Microsat Manager: An Application For Multi-locus Genetic Data, User's Guide and Documentation (Version 2.4)

J.R. Candy, C.G. Wallace, and T.D. Beacham

Fisheries and Oceans Canada
Pacific Biological Station
3190 Hammond Bay Road
Nanaimo, B.C.
V9T 6N7

2002

Canadian Technical Report of Fisheries and Aquatic Sciences 2407

Fisheries and Oceans Canada | Pêches et Océans Canada

Canada

Canadian Technical Report of Fisheries and Aquatic Sciences

Technical reports contain scientific and technical information that contributes to existing knowledge but which is not normally appropriate for primary literature. Technical reports are directed primarily toward a worldwide audience and have an international distribution. No restriction is placed on subject matter and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Technical reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1 - 456 in this series were issued as Technical reports of the Fisheries Research Board of Canada. Numbers 457 - 714 were issued as Department of the Environment, Fisheries and Marine Service Technical Reports. The current series name was changed with report number 925.

Technical reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport technique canadien des sciences halieutiques et aquatiques

Les rapports techniques contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui ne sont pas normalement appropriés pour la publication dans un journal scientifique. Les rapports techniques sont destinés essentiellement à un public international et ils sont distribués à cet échelon. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports techniques peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports techniques sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annual des publications scientifiques et techniques du Ministère.

Les numéros 1 à 456 de cette série ont été publiés à titre de rapports techniques de l'Office des recherches sur les pêcheries du Canada. Les numéros 457 à 714 sont parus à titre de rapports techniques de la Direction générale de la recherche et du développement, Service des pêches et de la mer, ministère de l'Environnement. Les numéros 715 à 924 ont été publiés à titre de rapports techniques du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 925.

Les rapports techniques sont produits à l'échelon regional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

102-04900/

Canadian Technical Report of
Fisheries and Aquatic Sciences 2407

2002

MICROSAT MANAGER: AN APPLICATION FOR MULTI-LOCUS GENETIC DATA, USER'S GUIDE AND DOCUMENTATION (VERSION 2.4)

by

J.R. Candy, C.G. Wallace, and T.D. Beacham

Fisheries and Oceans Canada
Pacific Biological Station
3190 Hammond Bay Road
Nanaimo, B. C.
V9T 6N7

© Her Majesty the Queen in Right of Canada, 2002
as represented by the Minister of Fisheries and Oceans.
Cat. No. FS 97-6/2407E ISSN 0706-6457

Correct citation for this publication:

Candy, J.R., C.D.Wallace, and T.D. Beacham. 2002. Microsat Manager: An application for multi-locus genetic data, user's guide and documentation (Version 2.4). Can. Tech. Rep. Fish. Aquat. Sci. 2407 : v + 113 p.

TABLE OF CONTENTS

	page
LIST OF TABLES	iv
LIST OF FIGURES	iv
ABSTRACT	v
RESUME	v
1. Introduction	1
2. Installation and Use	
2.1 Install MM2.4	2
2.2 Overview	2
3. Users Guide	
3.1 Main Menu	3
3.2 Formatting and importing data files	4
3.3 Setting up import formats and paths	5
3.4 Customizing import specifications	5
3.5 Screening data	6
3.6 Export data	6
3.7 Binning data	6
3.8 Baseline	9
3.9 Loci table	10
3.10 Mixture	10
3.11 Output choices	12
3.12 Example data	14
4. Input and Export formats	
4.1 Input format	14
4.2 Output formats	16
4.2.1 Spam	16
4.2.2 Bayes	17
4.2.3 Phylip	18
4.2.4 GDA	18
4.2.5 Genepop	19
4.2.6 Geneclass	19
4.2.7 Structure	20
5. Object Documentation (For Programmers)	
5.1 Tables	20
5.2 Queries	21
5.3 Macros	21
5.4 Modules	21
REFERENCES	24
ACKNOWLEDGEMENTS	25
Appendix A. Query SQL code	26
Appendix B. Form properties	27
Appendix C. Visual Basic Code in Modules	33
Appendix D. Wood Algorithm for binning alleles	102

LIST OF TABLES

Page

Table 4.1 Fixed column width baseline data example..... 14
Table 4.2 Fixed column width mixture data example..... 15
Table 4.3 Atlantic Cod names file..... 15
Table 4.4 Chinook salmon namesmix file..... 16
Table 4.5 Baseline output for Spam (allele counts)..... 16
Table 4.6 Mixture output for Spam..... 17
Table 4.7 Baseline output for Bayes (allele counts)..... 17
Table 4.8 Mixture output for Bayes..... 18
Table 4.9 Population file for Phylip..... 18
Table 4.10 Population file for GDA..... 19
Table 4.11 Population file for Genepop..... 19
Table 4.12 Population file for Structure..... 20
Table 5.1 List of tables and table descriptions..... 21
Table 5.2 List of queries and query descriptions..... 22
Table 5.3 MM2.4 macros..... 23
Table 5.4 List of main modules and descriptions..... 24

LIST OF FIGURES

Figure 3.1 Main Menu with species selection pull-down window..... 4
Figure 3.2 Edit Paths and files window..... 5
Figure 3.3 Manual Bin window..... 7
Figure 3.4 Autobin window..... 8
Figure 3.5 View Frequencies showing autobin allele counts..... 8
Figure 3.6 Baseline population selection window..... 9
Figure 3.7 Mixture selection window..... 11
Figure 3.8 Sub-event window showing Julian date selection 12
Figure 3.9 Output options for Spam baseline..... 13
Figure 3.10 Screening option for Geneclass output..... 13

ABSTRACT

Candy, J.R., C.D. Wallace, and T.D. Beacham. 2002. Microsat Manager: An application for multi-locus genetic data, user's guide and documentation (Version 2.4). Can. Tech. Rep. Fish. Aquat. Sci. 2407: v + 113 p.

MICROSAT MANAGER 2.4 is a relational database for multi-locus genetic data. It provides a Windows driven interface for queries, macros, and code that loads and manipulates genetic data. Data are loaded from either known populations (baseline) or unknown population aggregates (mixtures) and can be output in formats compatible with a number of statistical packages. The user can switch between species and output genetic data for estimating relative contributions of discrete populations in a mixture of populations (mixed stock analysis) using maximum likelihood estimation (Spam) or Bayesian techniques (Bayes), as well as other programs for genetic analysis (Genepop, GDA, Geneclass, Phylip, and Structure). MM2.4 provides options for binning data and adjustments to allele counts, to reduce "rare allele" effects in maximum likelihood estimations. This database is expandable to provide additional data manipulation and formatted output.

RÉSUMÉ

Candy, J.R., C.D. Wallace, and T.D. Beacham. 2002. Microsat Manager: An application for multi-locus genetic data, user's guide and documentation (Version 2.4). Can. Tech. Rep. Fish. Aquat. Sci. 2407: v + 113 p.

MICROSAT MANAGER 2.4 est une base de données relationnelles pour données génétiques sur plusieurs locus. Cette application est dotée d'une interface Windows qui accueille des requêtes, des macros et le code de chargement et de manipulation de données génétiques. Les données sont chargées à partir de populations connues (conditions de base) ou de regroupements de population inconnus (mélanges) et sont fournies en sortie dans des formats reconnus par plusieurs logiciels statistiques. L'utilisateur a le loisir de passer des données sur les espèces aux données génétiques de sortie et peut estimer les combinaisons relatives des populations discrètes dans un mélange de populations (analyse de stocks mélangés) en se servant de l'estimation du maximum de vraisemblance (Span) ou des techniques bayésiennes (Bayes), ainsi que d'autres programmées propres à l'analyse génétiques (Genepop, GDA, Genclass, Phylip et Structure). MM2.4 possède des options pour la mise en lots des données et l'ajustement des nombres d'allèles, qui peuvent réduire les effets « d'allèle rare » dans les estimations du maximum de vraisemblance. Cette base de données est extensible et peut prendre en charge des manipulations de données et des sorties formatées additionnelles.

1. Introduction

Recognition that isolated spawning populations may exist as independent evolutionary units has made the detection and protection of these populations increasingly important to fisheries management (Waples et al. 1990). Genetic databases are now used to refine our understanding of genetically distinct populations as well as provide natural markers for broodstock selection, forensics, and mixed stock analysis. The rapid expansion in the utility of genetic data has lead to development of new analytical programs (Luikart and England 1999) and the need for new data handling tools. To that end, relational database software can reduce both the burden of writing custom software applications and provide efficient and reliable means of handling large amount of multi-locus genetic data. A relational database provides a documentation system as well as an analytic tool. Microsoft Access our relational database of choice, is considered a standard in the Canadian federal government and used worldwide.

MICROSAT MANAGER 2.4 (MM2.4) provides a Windows interface to for loading, manipulating, and exporting combined multi-locus microsatellite, MHC (major histocompatibility complex), and/or allozyme data using built-in queries, forms, macros, Structured Query Language (SQL), and Microsoft Visual Basic for Applications (VBA). These datasets are inherently dynamic, so MM2.4 easily incorporates new species or new data for species already in the database. MM2.4 can be extended to provide additional data manipulation and formatting for new statistical analytical techniques.

This application allows automated data up-load from storage directories filling tables with allele counts of individuals sampled from either known populations (baseline) or unknown population aggregates (mixtures). Accompanying the genetic data at up loading are lookup tables containing names information associated with the particular species currently loaded. By selecting different options the user can determine binning levels, include or exclude loci, and include or exclude baseline populations and mixture data. Numerous output and data screening options are available within this application.

Section 2 discusses installation and use of MM2.4 providing a brief overview of the database structure. This section requires only minimal knowledge of Microsoft Access. Section 3 provides a users guide describing the Windows (Forms) interface. Section 4 examines the input and output file structures currently available. Section 5 gives details of different objects in the database, the object properties, and their interactions. Capital letters are used in the text (e.g. MAKE MIX GROUP) to identify control objects such as window names, pull-down windows, and command buttons. Italicized names are used in the text (e.g. *importspecs)* to identify other computer object such as tables.

2. Installation and Overview

System requirements:

Pentium III, 1 GHz (minimum)
128 MB Ram
Microsoft Access 2000

2.1 Install MM2.4

Step 1.

Download the file MM24.MDE from http://www-sci.pac.dfo-mpo.gc.ca/aqua/pages/bgsid.htm coping it to a directory on your harddrive.

Step 2.

Open the file using Microsoft Access 2000.

Step 3.

Before using MM2.4 you must set *Import* and *Export* paths. If the data sources are on different directories, these should be combined on a single directory for import. Only one import path can be used per species. Export paths must be specified to direct output to the appropriate data folders for statistical analysis (See Section 3.3 Setting up import and export formats and paths).

Step 4. [Optional]

Upon opening the database, the application will display the title page for a few seconds before displaying the MAIN MENU. Use the DATABASE VIEW button on the MAIN MENU to display the typical Access interface.

2.2 Overview

Readers should have basic familiarity with Microsoft Access. We briefly overview the structure of this application, but for in-depth information on specific questions, refer to **Help (F1** key) in Access.

An MDE file is a compacted, compiled Access database that has all the modules, with no editable source code. The modules run but cannot be viewed or edited. The size of the database is smaller with the source code removed and memory usage is optimized for improved performance.

Data are stored in either temporary (removed and re-loaded) or permanent (remains in the application until edited or deleted by the user) tables. Queries (graphical representations of SQL code) perform tasks on other database objects usually tables, and other queries. In this application the queries are called by VBA code in modules which the user initiates from the forms. The query SQL code is given in Appendix A. MM2.4 presents different forms for users to call tables, queries, other forms, macros, and modules (see Appendix B). Forms can be created from a number of different objects including text boxes, other forms, combo boxes, list boxes, command buttons, and sub-forms. Macros provide an intermediate between the forms and modules, allowing a form to initiate a number of tasks with one call. The modules contain a number of functions written in VBA code, which in this application usually start by opening tables and queries to provide data for the task at hand (see Appendix C). As mentioned above, neither forms

nor modules can be edited in the MDE file. Queries often execute more slowly then VBA code, so in this application VBA code is used in forms and modules to increase speed and reduce database complexity. Click on the forms tab and the MAIN MENU form to open the MAIN MENU form again.

3. Users Guide

This users guide is intended to help a new user become familiar with the features of MM2.4. An abbreviated version of this users guide can be seen in the README window on the MAIN MENU within the application.

3.1 Main Menu

The MAIN MENU is the starting point for any of the application operations. On starting MM2.4, the MAIN MENU pops up after several seconds of the title page. The MAIN MENU provides the user with a pull-down window for species selection, data handling parameters (UTILITIES), baseline selection/output (BASELINE), and mixture selection/output (MIXTURE). Close the MAIN MENU by clicking on the "Database View" button, then the DATABASE WINDOW with a list of tables is displayed behind this form.

The SPECIES selection pull down window on the MAIN MENU allows the user to switch between species (Figure 3.1). Selecting a species deletes all previous uploaded data tables, and creates empty data tables for each locus of the selected species, preparing for data input. Parameter settings (loci names, paths, binning, base/mix groups) are not deleted. See "Setting up import formats and paths" for adding new species to MM2.4.

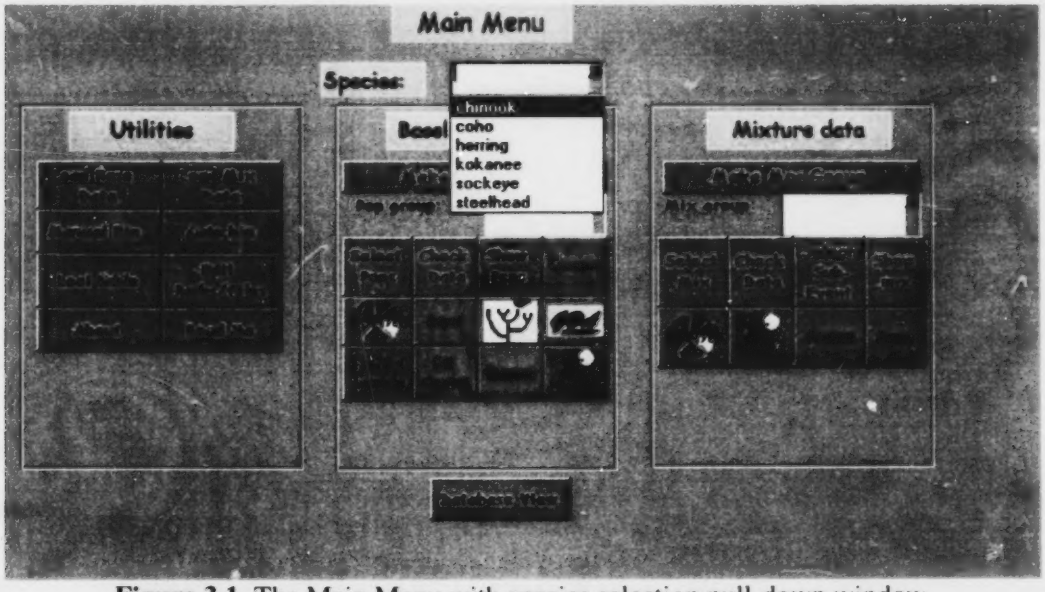

Figure 3.1. The Main Menu with species selection pull-down window.

3.2 Formatting and importing data files

Two types of data are imported to tables with each upload: genetic data (baseline and mixture files) and look-up tables providing names information for the genetic data. Baseline and mixture data are imported by individual loci from fixed field width text files (see Section 4 for examples of input data and format information). Fixed field widths settings can be changed to other import formats, see "Customizing Import Specifications".

Baseline data consists of tray#, population code, year sampled, individual number, and allele counts (0's, 1's, and 2's). Mixture data consists of tray#, sampling-event, (e.g. fishing gear/statistical area code), sub-event (e.g. Julian date of catch), individual number, and allele counts (0's, 1's, and 2's). Combinations of population/year/individual and sample-event/individual form a unique key for merging multiple loci in baseline and mixture data.

Each species has its own *names* and *namesmix* files which are imported at each data load. The names files consist of population name, regional population grouping code (i.e. run types, conservation units (ESU), geographic location), and population code. The namesmix files consist of sample event name, sample event code. See section 4 for further discussion on import formats.

3.3 Setting up import formats and paths

To edit the parameters used for importing the data open the EDIT PATHS/FILES button in the UTILITIES panel of the MAIN MENU. Then open the FILES AND PATHS window. Once established, the import parameters remain in the application until edited or deleted by the user (Figure 3.2). Fill in or edit the following fields in the FILES AND PATHS table:
Paths added in species list form
datatype: either mix or base
loci name: alphanumeric code (e.g. ots103)
importspecs: species name_loci (e.g. sock_ots103)
import files: file name on data directory (e.g. bse_ots103.txt or mix_ots103.txt)
alleles: number of alleles in import data.

These input parameters must be meaningful and must not be duplicated across tables; the loci names are used in the application's tables and output. Separate records for mixture and baseline files must be filled out in the FILES AND PATHS form, however the same *importspecs* should be used for both to ensure mixed and baseline data are imported using the same format when appropriate. See "Customizing import specifications" for more information on *importspecs*.

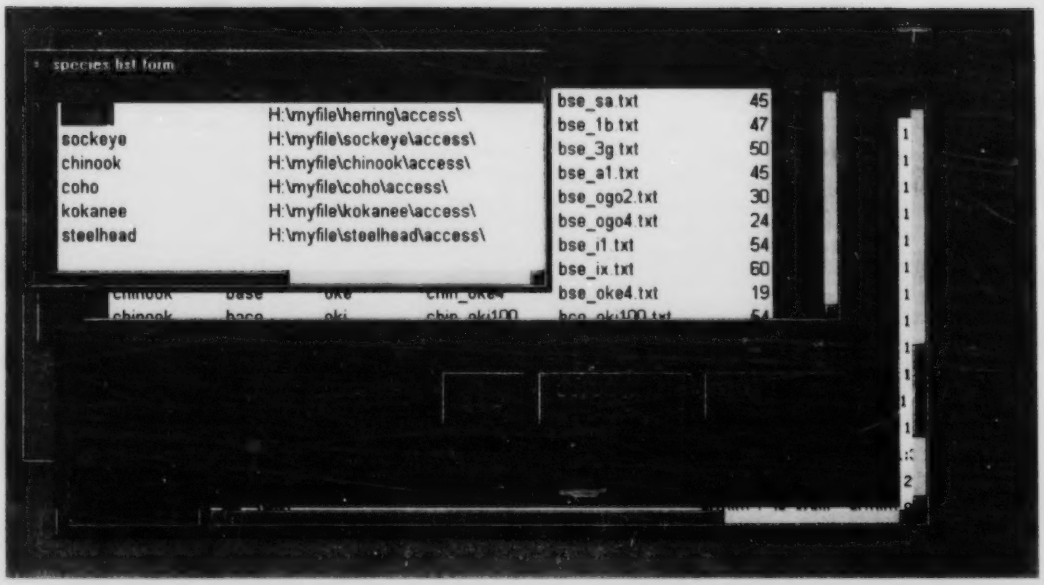

Figure 3.2 Edit Paths and Files window.

3.4 Customizing import specifications

Import specifications are stored in the hidden system tables *MsysIMEXColumns* and *MsysIMEXSpecs* and provide the import formats for the data including field widths, start columns and data type. To add or edit *importspecs* requires these system tables be accessed by clicking on FILE then GET EXTERNAL DATA then IMPORT. The user must browse to data file from storage directories. Once IMPORT TEXT WIZARD window is open go to ADVANCED. *Importspecs* can be changed here to use comma delimited or some other import format specification. Import specs can be seen but not edited by click on EDIT PATHS/FILES then click on VIEW SPECS. The *importspec* name must match that given in the FILES AND PATHS table.

3.5 Screening data

Records coded with tray "999" are removed after import into the application. This is useful for removing intentionally duplicate samples, which occur if a series of individuals were analyzed twice to test repeatability of allele sizing.

3.6 Export data

Output paths can be established by clicking on EDIT FILES AND PATHS, then clicking GO TO EXPORT PATHS. In this window the "application" field must be filled with "Spam" for Spam, "Bayes" for Bayes, "Geneclass" for Geneclass, "Phylip" for Phylip, "GDA" for GDA, and "Structure" for structure, followed by the path to the each application's directory. For proper output the user must use one of these names to be recognized by MM2.4.

3.7 Binning data

Binning data allows the user to reduce the effects of "rare alleles" on the maximum likelihood estimate, therefore binning only occurs to mixtures and baselines for SPAM. For all

other outputs the data remain unbinned. Population selection and output will not proceed for Spam unless bin ranges have been established for each locus.

To bin data, alleles with low baseline frequencies are combined with adjacent alleles. Known composition mixtures and simulations should be used to evaluate the effect of binning data. The MANUAL BIN button on the UTILITIES panel allows the user to set binning manually for individual loci or provides automated binning using the AUTOBIN button. To add or change allele binning open MANUAL BIN, select species then loci from pull-down menus, either add new allele ranges or edit allele ranges (Figure 3.3). Changes to allele binning are saved in the application until edited or deleted by the user.

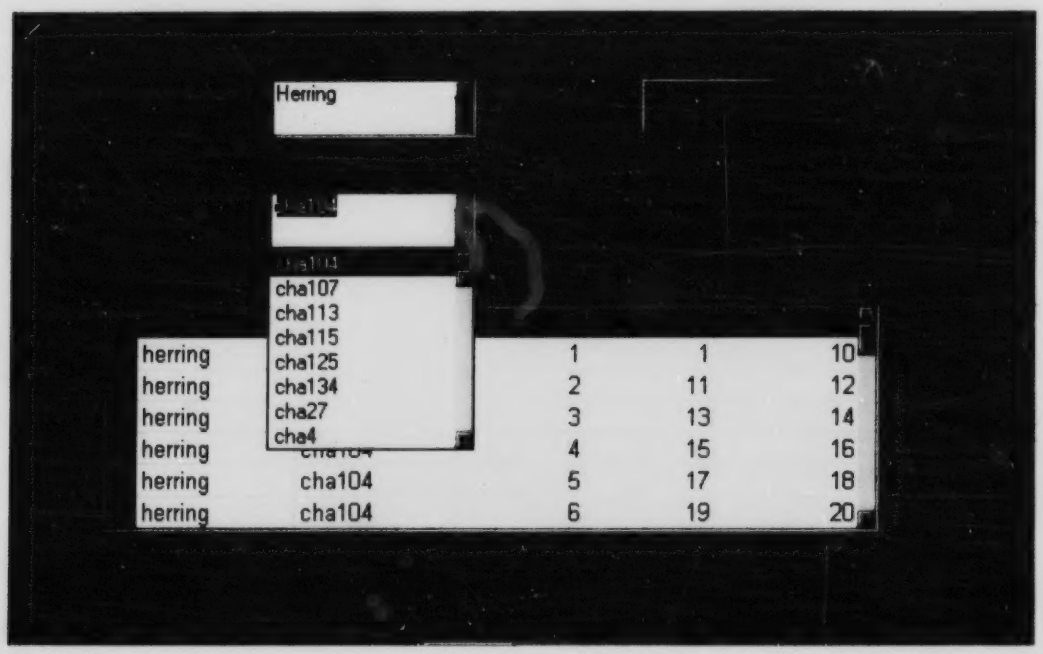

Figure 3.3 Manual Binning window.

To run automated binning, first run the selected baseline, see section on "baseline". Then click AUTOBIN button, set minimum bin threshold, click on RUN AUTOBIN (Figure 3.4). If no binning is required then select autobinning and set minimum bin frequency to a small value (<0.0001). MM2.4 defaults to manual binning during data upload and manual bin ranges are saved in MM2.4 between data uploads. Autobining will only remain in the application until a new data upload. New autobin bin ranges are established again by setting the minimum theshold value on the selected baseline.

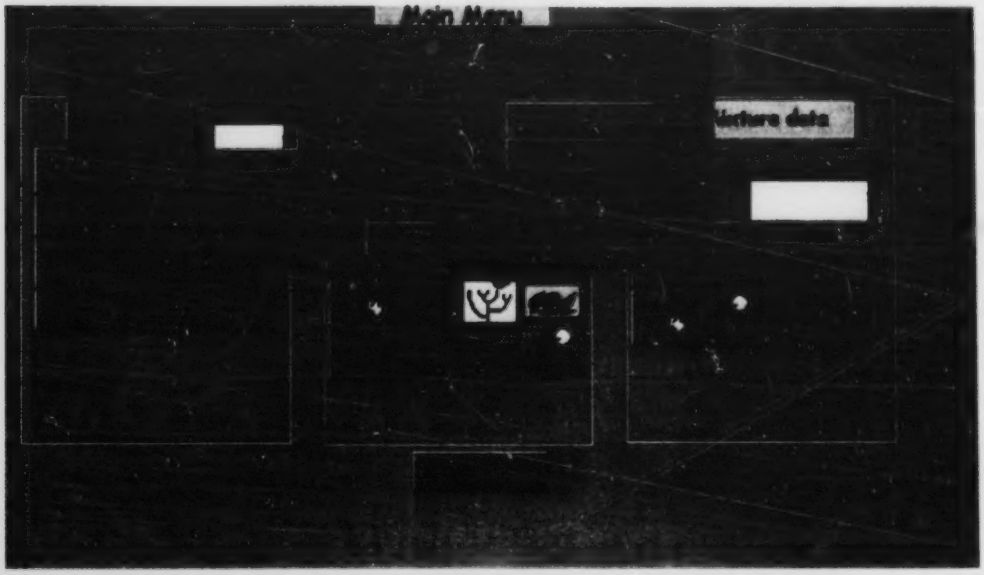

Figure 3.4 Autobin window.

Automated binning first determines the allele frequencies by locus for the selected baseline. For each loci it starts at the lowest frequency allele, binning adjacent alleles, right then left, in order to exceed a minimum bin frequency (Wood algorithm, C. Wood, pers. comm.). See Appendix D for details on this algorithm. This algorithm tends to bin the tails of allele frequencies heavily. Figure 3.5 shows the VIEW FREQUENCIES window with graph of binned allele frequencies.

Figure 3.5 View Frequencies window showing autobin allele counts.

3.8 Baseline

Baseline data consists of individual allele counts from known populations. LOAD BASE DATA on the UTILITIES panel initiates upload of baseline files, filling tables using the paths and import formats described above. A common error on upload is having the number of alleles in the empty loci table not match the number of alleles in the data files. Check allele numbers in EDIT PATHS/FILES and *importspecs* with data files.

The baseline data is loaded for the species selected in the SPECIES window of the MAIN MENU. Reselect species on SPECIES drop-down box if you wish to re-load data. The status bar on the bottom of the MAIN MENU shows the progress of the uploading loci.

The BASELINE panel lets the user select populations to build a baseline and choose output formats. MAKE POP GROUP opens forms to allow population selections from the *names* file (Figure 3.6). Select current species on the SPECIES window of MAKE POP GROUP. ADD/EDIT GROUP NAMES allows the user to provide a new name for the population grouping or change populations in a grouping already established. Population groups are kept in the *main_aggregate* table between uploads of current species. Use the < and > buttons to include or exclude individual populations from the grouping; or use >> and << to include or exclude all the populations from the grouping.

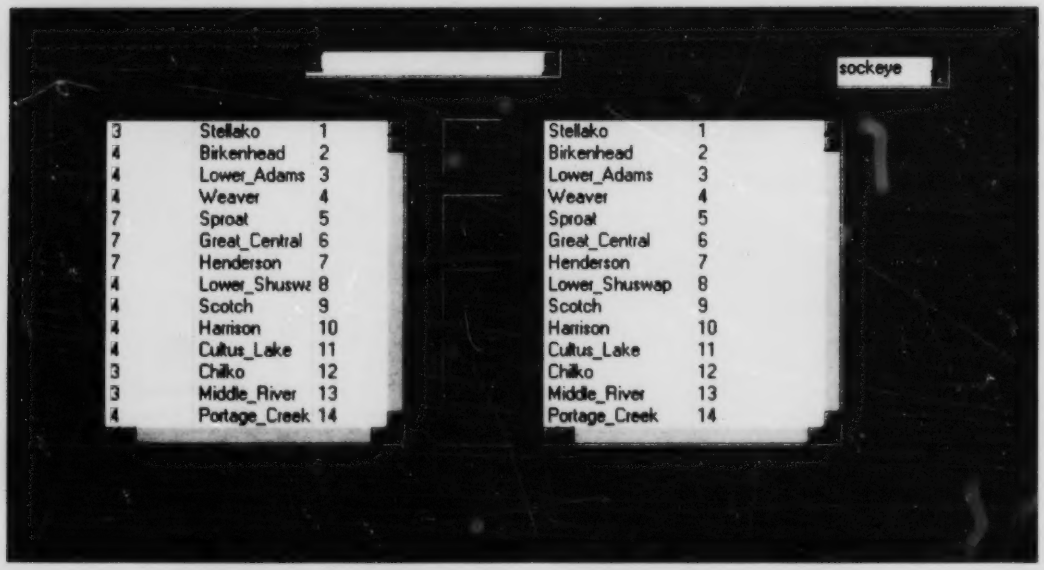

Figure 3.6 Baseline population selection window.

Closing MAKE POP GROUP and using the pull-down POP GROUP window lets user select a population grouping to be used for baseline build. SELECT POPS button builds a data table merging all loci by individuals using the selected populations. This button also builds a new LOCI TABLE and loads manual binning for current species. See "Loci Table" below for further description. SHOW POPS summarizes the current baseline selection, reporting the number of individuals per population per locus. This is useful if you plan to use the program Phylip, which will crash if loci are missing for some populations. Also use SHOW POPS to ensure that all the

required populations are present for output. Duplicate individuals or individuals with single alleles from the selected baseline are reported by <>2 ALLELES button and are screened out of the outputs. These data errors should be corrected in raw data files and reloaded before creating an output file.

3.9 Loci table

The LOCI TABLE button on the UTILITIES panel opens the *loci* table which provides summary information for each locus. This table consists of loci name, locus code (sequential number for locus), H-W (Hardy-Weinberg equilibrium) assumed or not (default is "yes"), the number of alleles at import (unbinned), and the number of binned alleles. As mentioned above, each time the SELECT POP button is clicked a new *loci* table is built from the baseline data. All outputs, baseline and mixture data, are restricted to those loci available in the loci table. Deleting records of loci from the *loci* table prevents output of these loci from both mixture or baseline formats. This table is not deleted until a new baseline is made. This gives the user the option of reloading mixture data without having to rebuild the *loci* table. For outputing phenotype data for Spam the H-W field should be set to "no". "Phenotype" will be printed in the control file instead of "locus".

3.10 Mixture

Mixture data consists of allele counts from individuals from unknown population aggregates. LOAD MIX DATA on the UTILITIES panel initiates upload of mixture files using paths and import formats described above. The status bar at the bottom of the MAIN MENU shows the progress of data upload. Sometimes not all loci are run for mixture analysis so a pop-up window informs the user which loci are unavailable for upload. The MIXTURE panel allows the user to import the current mixture files and select sample-events for analysis. MAKE MIX GROUP opens form to allow sample-event selections from the mix names files (Figure 3.7). Select current species in species window of MAKE MIX GROUP. ADD GROUP NAMES allows the user to provide a new name for the mixture grouping or edit existing grouping. Use the < and > buttons to include or exclude sample-events from the group.

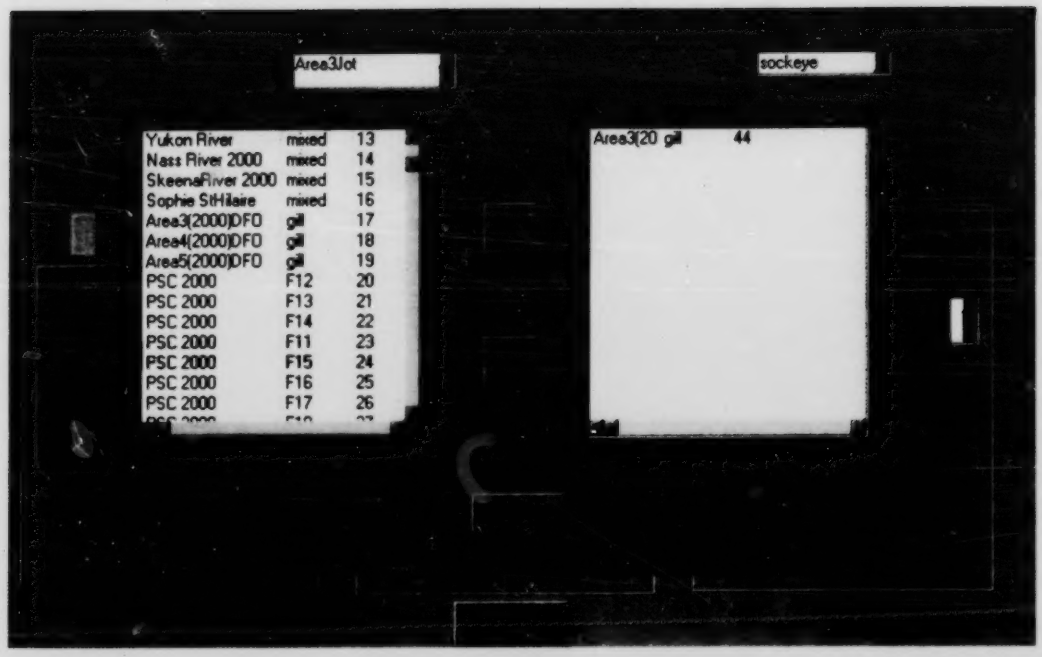

Figure 3.7 Mixture selection window.

Mixture groups are kept in the *main_aggregate mix* table between uploads of the current species. Once this window is closed, the MIX GROUP pull-down window lets the user select a mixture grouping to be used for mixture export. SELECT MIX builds a data table merging all loci using the selected mix group. The SUB-EVENT form must be used next to subset the data for select range (Figure 3.8). In most cases for fisheries applications this will be a range of Julian dates for a fishery. Click on MAKE RESTRICTED TABLE to select mixture data occurring between minimum and maximum dates. SHOW MIX provides a summary of the current mixture selection reporting the number of individuals per sampling-events per loci. Duplicate individuals or individuals falling outside the bin ranges from the selected mixture are reported by <>2 ALLELES button and are screened out of the outputs.

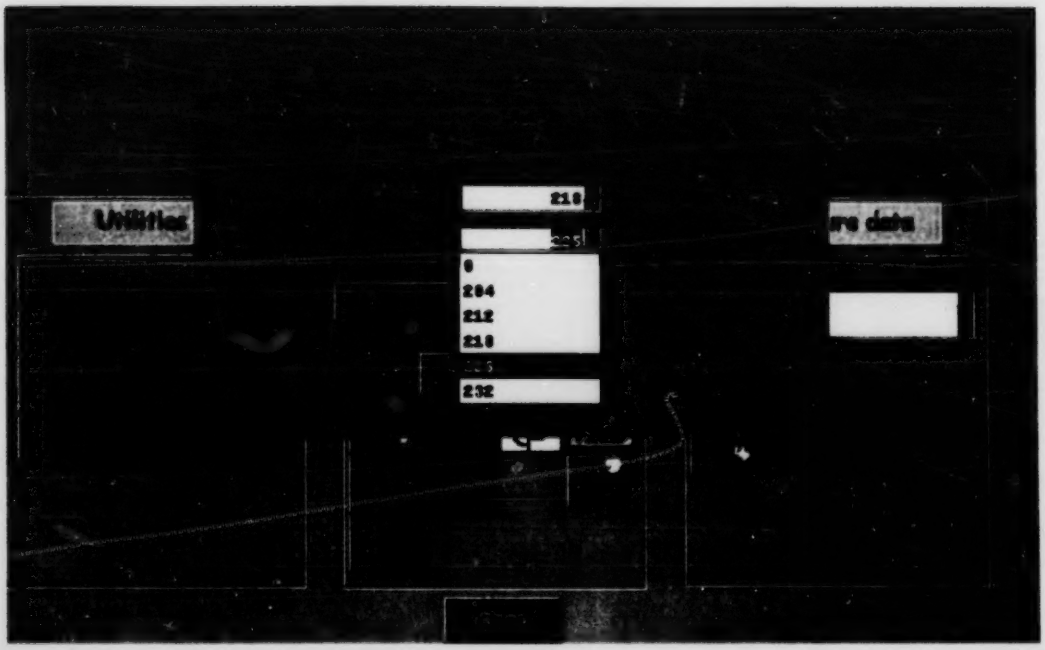

Figure 3.8 Sub-event window showing Julian date selection.

3.10 Output choices

For Spam only both baseline and mixtures are binned according the binning requirements established in the UTILITIES panel. Sort order for loci names in baseline, mixture, and control files follow the internal Microsoft Access sort order of first increasing numeric followed by increasing alpha characters in the first position of the loci name. Mixture analysis control files (simulation/estimation for Spam) are output with user-selected populations, loci, allele counts, and data paths. These control files provide a starting point for building a working control file; additional information may be required For example, the Spam simulation files will not contain the initial population contributions. A generic data file are output for both simulation and estimation (see Spam manual for control file options). To output formatted data, click on an application in the BASELINE panel. For Spam baseline files, a pop-up window (see figure 3.9) provides the user with an option of formatting output either as frequency data with "rare allele" adjusted (Rannala and Mountain 1997) or unadjusted count data. For discussion on this topic go to
http://www.cf.adfg.state.ak.us/geninfo/research/genetics/Software/Download/faqs/samplingzero.htm

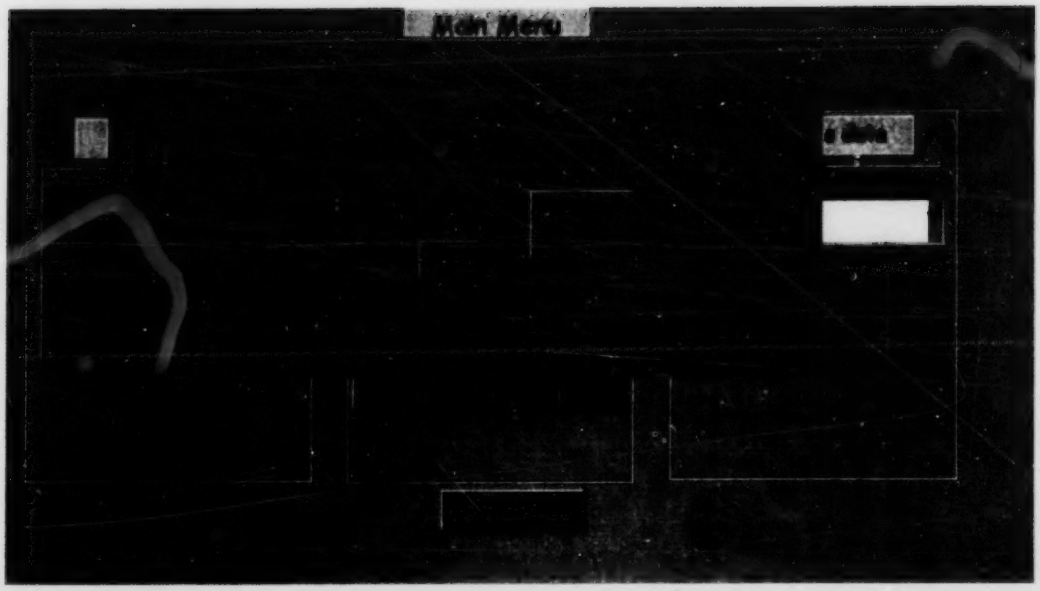

Figure 3.9 Output options for Spam baseline.

When clicking on output for GDA, Structure, and Geneclass a window allows the user to screen out individuals whose number of scored loci are below a certain threshold. This option exists for both the baseline and mixture data. Figure 3.10 shows an example for Geneclass output.

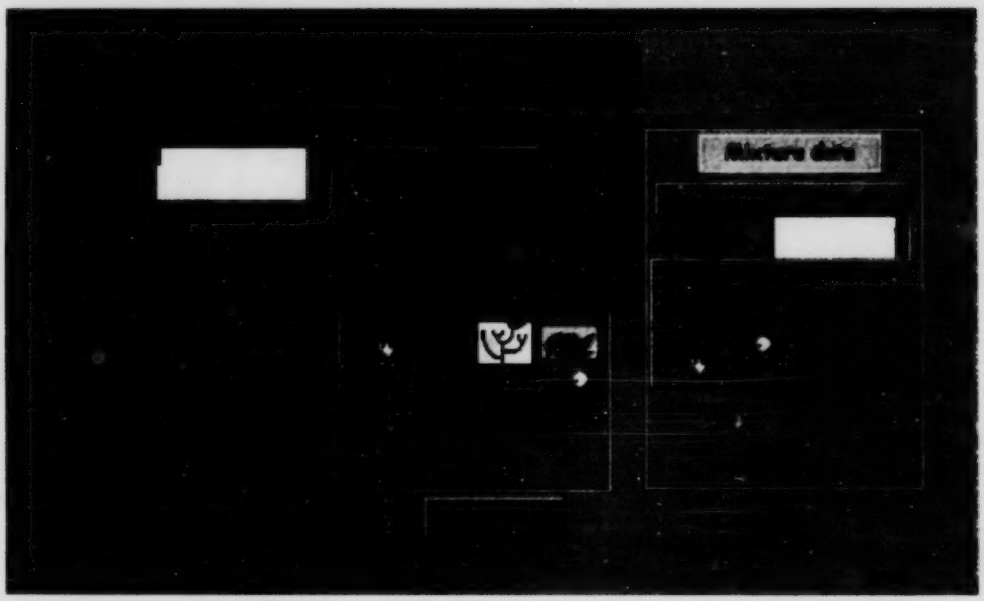

Figure 3.10 Screening option for Geneclass output.

3.11 Example data

MM2.4 provides sample data, consisting of three kokanee populations for use in exploring MM2.4. Import specs have already been established for this sample data (see EDIT FILES/FILES then VIEW SPECS). For the baseline data, change the paths in the FILES AND PATHS tables to the paths on your computer. Select 'kokanee' on the SPECIES selection window of the MAIN MENU to build empty loci tables. Click on LOAD BASE DATA to import the kokanee data. Click on MAKE POP GROUP to select populations for a baseline. Provide a new MAKE POP GROUP name and select several populations, then close the window. Build tables of the selected populations by selecting POP GROUP then SELECT POPS. See LOCI TABLE for a data summary. Binning has already been established for the kokanee and can be seen in the MANUAL BIN window. Go back to the BASELINE panel and see the selected baseline by clicking on SHOW POPS. Data can be output to any of the applications by clicking on the available buttons. The procedure for mixture data is similar to the baseline data, except that the SUB-EVENT must be selected before data is output.

4. Input and Output Formats

4.1 Input Format

Before data is imported, pre-processing is required to convert molecular weights into discrete allele counts. This procedure is not described here. Baseline data of fixed column widths consists from left to right of tray (gel), population code, sampling year, individual number, and allele counts (0,1,or 2's)(Table 4.1).

```
7  1 1992   1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
7  1 1992   2 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   3 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   4 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   5 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
7  1 1992   6 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   7 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   8 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
7  1 1992   9 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
7  1 1992  10 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992  11 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992  12 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992  13 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
7  1 1992  14 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0
```

Table 4.1 Fixed column width baseline data example.

Similarly, the mixture data columns are tray, sampling-event code (gear-area), sub-event code (Julian date), individual number (fish number), and allele counts (0,1,or 2's)(Table 4.2.)

1	83	276	10000000000000200000
1	83	277	20000100000000010000
1	83	277	31100000000000000000
1	83	277	41000000000010000000
1	83	277	50100000000000100000
1	83	278	60010100000000000000
1	83	278	80000200000000000000
1	83	278	90000000000020000000
1	83	279	10000000000000000000000
1	83	279	11000000000000000000000
1	83	279	12010000000000000001000
1	83	280	13100000000000100000000
1	83	280	14000010001000000000000

Table 4.2 Fixed column width mixture data example.

A unique names file is associated with each species imported into the application. The names file consists of the following columns: population name, a regional code, population code number, and a regional name corresponding to the population code number (Table 4.3).

Off_2J	02	1	1=Flemish Cap
Funk_Island	02	2	2=NE offshore
Notre_Dame	03	3	4=3Ps offshore
Flemish_Cap	01	4	3=NE inshore
Grand_Bank	02	5	5=3Ps inshore
Trinity_Bay	03	6	6=3Pn
Bonavista_Bay	03	7	7=3N
Gilberts_Bay	03	8	
Burgeo_Bank	04	9	
Halibut_Ch	04	10	
Fortune_Bay	05	11	
Placentia_Bay	05	12	
Hawke_Channel	02	13	

Table 4.3 Atlantic Cod names file.

A unique namesmix file is associated with each species imported into the application. Import namesmix files are similar to the names file but consist of the following columns: event name (in this case two columns sample location and sample type), event code, and region code (Table 4.4).

Sechelt Creel	Sport	36	1
Powell Creel	Sport	37	1
Vancou. Creel	Sport	38	1
Campbell Creel	Sport	39	1
Nanaimo Creel	Sport	40	2
IOS Toxicol	Known	41	2
Skeena test	Gill	42	2
Area1 Langara	sport	43	2
TopAgg(juv)	Trap	44	2
LowSumas(juv)	Trap	45	2
MidHarrison(ju	Trap	46	2
Area28	Sport	47	

Table 4.4 Chinook salmon namesmix file.

4.2 Output Formats

This section overviews the data output formats. MM2.4 can output data for a number of statistical packages available over the internet. Be sure to read the package's manual for more detailed information.

4.2.1 Spam

Spam (Debevec et al. 2000, Pella et al. 1996) uses maximum likelihood estimation methods to determine relative contributions of discrete populations in a mixture of several populations as mentioned in section 3. An executable file and documentation for Spam can be downloaded from the Alaskan Department of Fish and Game website: http://www.cf.adfg.state.ak.us/geninfo/research/genetics/Software/SpamPage.htm. Highly polymorphic genetic data should be binned down to avoid "rare alleles" effects in mixtures. Table 4.5 shows an example of MM2.4 output for a single baseline population where loci are reported as "locus" for the Spam analysis. In this example the baseline is output as allele counts instead of frequencies. Spam assumes loci names start with alpha character, numerics at the start must be changed in both control and baseline files. See Spam documentation on input file limits.

```
# 001 Stellako
a1b       74 0 0 591 79 54 0 0
a3dre     0 12 83 0 0 37 244 62 28 21 22 47 63 22 35 51 29 0 0 0
beta1     0 9 66 453 26 0 0 1 0 0 2 0 0 0 1
i1        0 1 320 88 0 75 37 4 131 2 27 0 23 3 58 1 10 1 1
oki10     0 0 0 1 1 1 5 5 15 28 20 22 86 40 43 92 36 67 35 22 3 6 36 67 68 37 12 0 18 0 0 0
oki16     0 0 0 0 0 0 1 0 5 354 33 1 0 38 72 12 6 39 32 1 0
oki1a     0 26 12 639 115
oki1b     210 37 502 17
oki29     0 0 15 200 61 30 48 2 4 28 2 8 63 38 9 82 10 27 44 15 0 0 16
oki6      12 304 3 188 1 214 4 15 3 0 0 0 0 0 0 0 0 0 0 0
omy77     0 1 190 0 85 75 5 107 2 17 150 33 75
one8      0 0 47 50 10 318 77 168 0 86 0
ots103    7 4 29 72 19 13 10 36 87 46 61 85 62 61 97 63 31 2 0 1
ots2      0 0 279 0 2 0 0 116 7 74 199 12 73 14 0
ots3      0 0 104 12 510 2 125 15 0 0 2 0
```

Table 4.5 Baseline output for Spam (allele counts).

Table 4.6 shows mixture file output from MM2.4. Each record contains all data for a single individual where each locus is represented by series of alleles counts; the last column is the individual's number. Missing data at a locus is output as all zeros. The dataset starts with "\" in first column. Loci in the mixture data must be read in to Spam in the same order found in the control file. MM2.4 internal sort will output both control, mixture, and baseline files in the same sort order with loci names output as increasing numeric followed by increasing character values in the first position of the name. Each column corresponds to an allele and each row corresponds to an individual; a white space separates loci.

```
\
0001000000010000000 0002000000000000 0000000000020000 0200000000000000 00020000 000010100000  1
0000000000000000000 0000000100010000 0000000000020000 0002000000000000 00110000 000000200000  2
0000001100000000000 0000100000001000 0001000000000100 0000000100000001000 10100000 000020000000  3
0000010010000000000 0000000000010100 0000000000020000 0000000011000000000 01010000 000011000000  4
1000000100000000000 0000000000000000 0000000000010010 0020000000000000000 00200000 000100100000  5
0000002000000000000 0000010000000100 0000000000101000 0000010000000010000 00110000 100100000000  6
0000000000100000001 0000000000000000 0000001000010000 0000100000000010000 00020000 000001100000  7
0000000000001001000 0001000100000000 0010000000010000 0000000100000001000 00200000 000100010000  8
0001000000001000000 0000100000000100 0001000001000000 0000010100000000000 00200000 000020000000  9
0000000001000010000 0000000000000000 0010000100000000 0000000110000000000 00020000 000000110000 10
0000100001000000000 0000000000000000 0000000000020000 0001000100000000000 00200000 000020000000 11
```

Table 4.6 Mixture output for Spam.

4.2.2 Bayes

Bayes (Pella and Masuda 2001) uses Bayesian methods to estimate relative contributions of discrete populations in a mixture of several populations. An executable file and documentation can be downloaded from a NOAA ftp directory ftp://ftp.afsc.noaa.gov/abl/sida/mixture-analysis/. Table 4.7 shows an example of baseline data for Bayes where columns from left to right consist of population identifier, loci identifier, number of alleles, and counts of alleles at a locus.

1	1	100	0	6	8	1	0	17	17	19	39	47	6	2	15	21	2	0
1	2	176	4	0	21	21	52	18	12	19	39	41	23	15	19	35	14	3
1	3	184	0	0	11	1	28	30	64	13	27	15	47	16	39	30	41	6
1	4	140	0	0	0	73	52	75	37	13	5	3	0	0	18	4		
1	5	141	3	13	19	30	1	1	12	15	10	31	79	43	1	24		
1	6	185	0	11	1	0	26	20	22	22	29	75	69	32	47	2	14	
1	7	169	0	18	1	2	4	48	39	6	39	26	5	40	74	26	1	9
1	8	150	0	160	80	25	21	13	0	1								
1	9	150	2	5	0	11	1	0	10	5	10	1	28	3	7	12	13	3
1	10	156	0	0	76	154	20	25	1	11	12	0	8	1	1	2	1	0
1	11	129	56	8	100	8	0	15	40	2	0	20	1	0	8			

Table 4.7 Baseline output file for Bayes (allele counts).

Bayes mixture files are similar in format to Spam, with each column corresponding to an allele and each row corresponding to an individual. Loci are separated by white space (see table 4.8). Bayesian analysis does not require that baseline allele frequencies adjusted for rare alleles.

000000000000002000 00110000 000100010000	115
000000000000001100 00200000 000000200000	125
000000000000002000 00002000 000100010000	180
000100000001000000 00200000 000200000000	182
000000010000000100 00101000 000000200000	186
020000000000000000 00200000 000101000000	195
000000100000000010 00200000 000200000000	200
001000000000010000 00200000 000000200000	201
000010010000000000 00110000 000100100000	202
000000000000000000 00101000 000100100000	211

Table 4.8 Mixture output file for Bayes.

4.2.3 Phylip

Phylip (Felsenstein 1993) is a series of programs for inferring phylogenies, determining genetic distances and drawing trees. Phylip can be downloaded from University of Washington website http://evolution.genetics.washington.edu/phylip.html. Header information consists of number of populations, number of loci, number of alleles per loci. The body of the data consists of population name and allele frequencies (Table 4.9). For each population this program parses allele frequencies by loci given the number of allele per loci in the header information; consequently all populations must contain all loci for the program to read the input file properly. The baseline can be checked using SHOW POPS on the MAIN MENU.

```
30        15
 14 28 15 33 75 23 8 9 35 30 20 43 30 24 24
Birkenhead 0.00531915 0.00000000 0.00000000 0.00000000 0.00000000
           0.00000000 0.00000000 0.00000000 0.00000000 0.00450450
           0.00210970 0.00421941 0.06118143 0.16033755 0.70042194
           0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
           0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
           0.00947867 0.00000000 0.00000000 0.01658768 0.00000000
           0.00000000 0.00000000 0.00369004 0.00369004 0.69372694
           0.00192308 0.00769231 0.39230769 0.03846154 0.52307692
           0.00000000 0.00869565 0.05652174 0.03478261 0.17391304
           0.00823045 0.00000000 0.13374486 0.00000000 0.00205761
           0.00371747 0.00000000 0.00000000 0.02230483 0.00000000
```

Table 4.9 Population file for Phylip.

4.2.4 GDA

GDA (Weir 1996) performs disequilibrium analysis, population structure, genetic distances, and inbreeding parameters. GDA can be downloaded from http://lewis.eeb.uconn.edu/lewishome/software.html. Data is output in nexus format; header information includes number of loci, number of populations, formats for the data and loci names. The body of the data consists of individual fish per record and corresponding alleles for each locus (Table 4.10).

```
#nexus
begin gdadata;
dimensions nloci=10 npops= 28;
   format missing=? separator=/;
   locusallelelabels
     1     3g,
     2    a1co,
     3    coga,
     4    oki1,
     5   oki10,
     6  oki100,
     7  oki101,
     8  ots103,
     9    ots2,
    10    ots3,
;
   matrix
Alouette      :
1995_201  15/27 09/17  ?/?   ?/?  22/26 29/65 03/05 05/29 04/05 14/15
1995_202  16/21 08/08  ?/?  12/14 05/25 17/17 04/05 16/21 04/05 15/15
1995_203  11/11 06/08  ?/?  09/10 14/17 41/55 06/15 14/14 04/05 12/13
1995_204  04/06 14/15  ?/?  10/13 17/25 13/17 04/07 05/32 04/06 14/17
```

Table 4.10 Population file for GDA.

4.2.5 Genepop

Genepop computes exact tests for H-W equilibrium, population differentiation, and genetic disequilibrium among loci, Fst, and allele frequencies. It can be downloaded from http://wbiomed.curtin.edu.au/genepop/. Genepop format consists of a header title, loci names then population records. Population records contain individuals with description of individuals separated from the data by a comma. Population are separated by "POP"(Table 4.11).

```
punt97
3g
i1
ix
sa
a1
1b
POP
Puntled(s)  105 1997  31 , 0817 0000 1527 0000 1324 1617
Puntled(s)  105 1997  32 , 0000 0000 1515 1924 0916 1416
Puntled(s)  105 1997  33 , 0522 0000 1927 2424 1116 0612
Puntled(s)  105 1997  34 , 1517 0000 0000 2425 1316 1417
```

Table 4.11 Population file for Genepop.

4.2.6 Geneclass

Geneclass (Cornuet et al. 1999) uses Bayesian analysis employing multi-locus genotypes to select or exclude populations as origins of individuals. Bayesian analysis does not

require that baseline allele frequencies adjusted for rare alleles.This program can be downloaded from http://www.ensam.inra.fr/URLB/. Geneclass has essentially the same format as Genepop so is not shown here.

4.2.7 Structure

Structure (Pritchard 2000) is a model-based clustering method using Bayesian analysis for multilocus genotype data to infer a populations structure and assign individuals to a population. This program can be downloaded from http://www.stats.ox.ac.uk/~pritch/home.html Data is arranged as a matrix in which data for an individual is stored as two consecutive rows. The first few columns are for non-genotype data, the first column contains information of individuals (population, year, and individual number). The second column is an integer designating a user-defined population from which the individual was obtained. The third column is a boolean flag indicating where or not to use the population information from the previous column when using the learning samples. The default for output is "1" which uses the population information. Missing data is designated by "-9" (Table 4.12).

```
04019940064   1 1 02 24 32 14 03 29 27 10 -9 10 18 11 22
04019940064   1 1 15 24 33 14 19 35 31 10 -9 11 21 11 33
04019940065   1 1 02 23 24 11 03 26 35 09 21 10 10 10 30
04019940065   1 1 07 26 25 12 14 29 50 10 24 11 30 12 35
04019940066   1 1 04 24 23 -9 18 29 37 -9 23 05 10 -9 25
04019940066   1 1 19 24 28 -9 19 30 50 -9 29 12 10 -9 35
04019940067   1 1 06 17 09 01 06 24 35 10 28 11 17 -9 06
04019940067   1 1 20 30 23 01 06 28 35 10 32 11 18 -9 28
04019940068   1 1 14 23 18 13 12 22 36 12 24 04 10 -9 29
```

Table 4.12 Population file for Structure.

5. Object Documentation (For Programmers)

MM2.4 uses a number of different object types, forms, queries, tables, macro, reports and modules to analysze and output data. This section catalogues the different object types and their properties. The windows (forms) are most familiar to the user since they provide the interface to MM2.4.

Tables

Tables provide either temporary or permanent data storage. Temporary tables contain genetic data which change with each data upload, while permanent tables remain in the application until edited or deleted by the user. Permanent tables contain import/export paths, base and mixture aggregates for each species, and binning strategies. Interface windows allow the user to run More of the data can be viewed through the interface windows, however sometimes the user will find it userful to check specific records in certain tables. Table 5.1 provides table descriptions.

Table 5.1 List of tables and table descriptions.

TABLE	TYPE	STATUS	CONTENTS
Bse_*	Visible	Temporary	Baseline allele counts by locus for a species
Mix_*	Visible	Temporary	Mixture allele counts by locus for a species
Names	Visible	Temporary	Names and name code file for baseline names
Namesmix	Visible	Temporary	Names and name code file for mixture names
main_aggregation	Hidden	Permament	Baseline groupings by species
main_aggregation mix	Hidden	Permanent	Mixture groupings by species
Lociprams	Visible	Temporary	Summary of current locus
Manautobin	Visible	Temporary	Bin ranges manual/auto for current species
Main_bin	Hidden	Permanent	Manual binning by species
ImportSpecTbl	Hidden	Permanent	Import files, import specs by locus by species
ExportSpecs	Hidden	Permanent	Export paths by statistical package
msysImexColumns	Hidden	Permanent	System file with import specifications
msysImexSpecs	Hidden	Permanent	System file with import specifications
all_base	Hidden	Temporary	Baseline of allele counts by individual all loci
all_mix	Hidden	Temporary	Mixture of allele counts by individual all loci
all_mix2	Hidden	Temporary	Subset of all_mix selected by sub-event
Temp	Hidden	Temporary	Current species and path
Species list	Hidden	Permanent	All species/paths capable of upload
00_readme	Hidden	Permanent	Help file

5.1Queries

Queries are the graphical equivelent of SQL code (See Appendix table A). MM2.4 uses select, cross-tab, make table, append, and delete queries. Typically these queries are linked in series prior to to passing manipulated data to modules for output (Table 5.2).

5.2Macros

Most of the queries are called directly from modules, however it can be useful to us a macro to initiate a number of actions at one. A list of macros and their corresponding function can be seen in Table 5.3.

5.3 Modules

Modules contain VBA code are called from forms. Typically a module will initiate a series of queries to provide data to be output by the module. A list of modules, input and output tables and queries and their corresponding functions can be in Table 5.4.

Table 5.2 List of queries and query descriptions.

NAME	TYPE	SERIES	FUNCTION
01_missingBsedata	Select/xtab	1-2	Report individuals per tray per locus
01_missingMixdata	Select/xtab	1-2	Report individuals per tray per locus
03_fish per locus	Select	1-2	Total number of individuals per population per locus
04_geno_nn	Select	1-6	Genotypes of base individuals ouput modules for neural net
04_geno_sp	Select	1-7	Genotypes of base individuals output modules for spam baseline
04_genomix_sp	Select	1-6	Genotypes of mix individual for output modules for spam mixture
04_julian day select	Select	1-2	Select sub-event from mix2 file
05_gda header	Select	1-3	GDA header for output modules for GDA (#loci, #populations)
05_phylip header	Select	1-2	Phylip header output modules (#loci, #pops, allele cnt)
05_phylip	Select	1-2	Phylip output to modules allele frequencies by stock
06_check regions	Select	1-3	Header information for spam estimate and simulation control files
07_gda	Select	1-3	Genotypes of base individuals for output modules for phylip
07_geneclassmix	Select	1-3	Genotypes of mix individuals for output modules to geneclass
07_genpop	Select	1	Query of gda base individuals for output modules genpop
09_binfreq	Select	1-4	Allele freq pre locus for select baseline used by autobin
10_bayesbase	Select	1-3	Allele counts by stock and loci for output modules bayes
11_allbase_sort	Select	1	Sort of loci
11_manautobin_allelecnts	Select	1	Max number of alleles and binned alleles from automanbin table
12_ListImportSpecs	MakeTable	1	Makes TempImportSpecs table from ImportSpecs and formComboListImportSpecs
13_manautobincounts	Select	1-3	Freq of allele by locus
14_showbase14_showmix	Select/xtabSelect/xtab	1-21-2	Shows umbers of individuals by population and locus in baseline dataShows Sub-Event mixture data
14_showmixbasescreen	Select/xtabSelect	1-21-3	Shows Sub-Event mixture dataFinds individual with more or less than two alleles from base file
basescreenDel_aggregate	SelectDelete	1-31	Finds individual with more or less than two alleles from base fileDelete population from main_aggregation selected by user
Del_aggregateDel_aggregate all	DeleteDelete	11	Delete population from main_aggregation selected by user Delete aggregate group from main_aggregation selected by user
Del_aggregate allmixscreen	DeleteSelect	11-5	Delete aggregate group from main_aggregation Finds indivduals with more or less than two alleles from mix file
Del_aggregate mix	Delete	1	Delete Event from main_aggregation mix selected by user
Del_aggregate all mixShow fish base	DeleteSelect/xtab	11-2	Delete aggregate group from main_aggregation mixSummary of baseline individuals per locus per population
Add all baselineShow fish baseShow fish mix	AppendSelect/xtabSelect/xtab	11-21-2	Adds population to baseline selection with grouping identifier Summary of baseline individuals per locus per population Summary of mixture individuals per loci per population
Add all mixtureShow fish mixSort bin ranges	AppendSelect/xSelect	11-21	Adds Event to mixture selection with grouping identifier.Summary of mixture individuals per loci per population Sorts loci from bin_ranges table
mixscreenShow fish baseShow fish mixSort bin ranges	SelectSelect/xtabSelect/xtabSelect	1-51-21-21	Finds indivduals with more or less than two alleles from mix file. Summary of baseline individuals per loci per population Summary of mixture individuals per loci per populationSorts loci from bin_ranges table
Show fish baseShow fish mixSort bin ranges	Select/xtabSelect/xtabSelect	1-21-21	Summary of baseline individuals per locusper populationSummary of mixture individuals per loci per populationSorts loci from bin_ranges table
Show fish mixSort bin ranges	Select/xtabSelect	1-21	Summary of mixture individuals per locus per populationSorts loci from bin_ranges table
Sort bin ranges	Select	1	Sorts loci from bin_ranges table

NAME	ACTION	CALLS
autobinning	Runcode,Rename	Fill tables(), rename autobin to manautobin tbl
autoexec	OpenFrm, CloseFrm,OpenFrm	Open title, Close title, Open MainMenu
Base summary	OpenQry,OpenQry	Basescreen(3)-remove fish, show fish base(2)
Help	OpenRpt	Helpme report
SelectJulianDate	Openquery	04 Julian day select(1)

Table 5.3 MM2.4 macros.

Table 5.4 List of main modules and description.

NAME	INPUT	OUTPUT	CALLS	FUNCTION
Import()	Temp, ImportSpecTable	Bse_*, Names	None	Imports baseline and names files
Importmix()	Temp, ImportSpecTable	Mix_*, Namesmix	None	Imports mixture and mixnames files
ChangePathBse()	MainMenu!species,temp, ImportSpecTable, main_bin	Temp, Bin_ranges (manautobin)	Create_bins()	Check previous species match current, delete old tables, build new tables to match current species, rename bin_Ragne to automanbin
SelectBaseStock()	Base_*, main_aggregation	All_base	Loci_prams() Append Tables()	Use aggregate group (special name) to select stocks into all_base
Autobin.Binning()	Autobin, Forms![Autobin Form]!Combo1	None	None	Bins adjacent allele of lowest frequency for each loci to a minimum threshold provided by user.
Autobin.FillTables()	09_BinFreq(4)	Autobin	Binning()	Builds new autobin tables
Autobin.ChangeFreq(...)	Autobin	Autobin		Modify Autobin table by change sequence of binned alleles.
Append.tables(In_table,Out_table,primer	Out_Table, In_Table	Out_Table,In_Table	None	Append additonal data to existing tables
BaseSelectAll()	names	test	None	unkown
Phylip_text()	05_phylip(3), 05_phylip header(3), ExportSpecs	Phylip.txt	None	Takes header information and allele frequencies per population and outputs in phylip format.
Genepop()	07_genopop(1) 05_gda header(2) ExportSpecs	Genepop.bse	None	Takes header information and alleles per individual and outputs in genepop format
Spam_est()	06_check regions(2),lociprams, ExportSpecs	Est.ctl	None	Takes select baseline population and region information, loci, allele counts and outputs spam estimation file
Spam_sim()	06_check regions(2), lociprams, ExportSpecs	Sim.ctl	None	Takes select baseline population and region information, loci, allele counts and outputs spam simulation file
Spam_bse()	04_geno_sp(6), lociprams, ExportSpecs	Spam.bse	None	Takes allele counts per population and outputs select baseline in Spam format Either "genos" or "phenos"
Spam_mix()	04_genomix_sp(5)	Spam.mix	None	Takes allele counts of individuals and outptuts in Spam format.

REFERENCES

Cornuet, J., Piry, S., Luikart, G., Estoup, A., and Soliqnac M. 1999. New methods employing multilocius genotypes to select or exclude populations as origins of individuals. Genetics 153:1989-2000.

Debevec, E.M., R.B. Gates, M. Masuda, J.Pella, J. Reynolds, and L.W. Seeb. 2000. SPAM (Version 3.2) Statistics program for analysis mixtures. J. Heredity 91(6).

Felsenstein, J. 1993. PHYLIP (Phylogeny Inference Package), version 3.5c Department of Genetics, University of Washington, Seattle.

Luikart G. and England P.R. 1999. Statistical analysis of microsatellite DNA data Tree 14:7 253-256.

Pella, J. and M. Masuda. 2001. Bayesian methods for analysis of stock mixtures from genetic characters. Fish. Bull. 99:151-167

Pella, J., M. Masuda, and S. Nelson. 1996. Search algorithms for computing stock composition of a mixture from traits of individuals by maximum likelihood. NOAA Tech. Memo. NMFS-AFSC-61 p. 1-68.

Pritchard, J.K., Stephens, M. and Donnelly, P. 2000. Inference of populations structure using multilocus genotype data. Genetics 155 : 945-959.

Rannala, B., and Mountain, J.L. Detecting immigration by using multilocus genotypes. Proc. Natl Acad. Sci. USA 94 :9197-9221.

Waples, R.S. Winans, G.A., Utter, F.M., and Mahnken, C. 1990. Genetic approaches to the management of Pacific salmon. Fisheries 15 :5 19-25.

Weir, B.S. 1996. Genetic data analysis II. Sinauer Associates Inc. Sunderland, MA. USA.

ACKNOWLEDGEMENTS

We would like to thank Michele Masuda, Alaska Fisheries Science Center, NOAA Juneau, AK for useful comments during the development of this software. We would also like to thank Joel Reynolds, Gene Conservation Lab, Alaska Department of Fish and Game, Anchorage, AK, Chris Wood, Department of Fisheries and Oceans, Pacific Biological Station, Nanaimo, B.C. for reviewing this manuscript and providing many helpful suggestions and editoral comments.

Appendix A. SQL code in Queries

01_missingBsedata(1)

```
SELECT all_base.tray, all_base.stock, all_base.year, all_base.Locus, all_base.fish AS
    Individual
FROM all_base
GROUP BY all_base.tray, all_base.stock, all_base.year, all_base.Locus, all_base.fish;
```

01_missingBsedata(2)

```
TRANSFORM Count([01_missingBseData(1)].Individual) AS CountOfIndividual
SELECT [01_missingBseData(1)].year, [01_missingBseData(1)].stock
FROM [01_missingBseData(1)]
GROUP BY [01_missingBseData(1)].year, [01_missingBseData(1)].stock
PIVOT [01_missingBseData(1)].Locus;
```

01_missingMixdata(1)

```
SELECT all_mix.tray, all_mix.stock, all_mix.year, all_mix.Locus, all_mix.fish AS
    Individual
FROM all_mix
GROUP BY all_mix.tray, all_mix.stock, all_mix.year, all_mix.Locus, all_mix.fish;
```

01_missingMixdata(2)

```
TRANSFORM Count([01_missingMixData(1)].Individual) AS CountOfIndividual
SELECT namesmix.Area, namesmix.Gear, [01_missingMixData(1)].tray,
    [01_missingMixData(1)].year, [01_missingMixData(1)].stock
FROM namesmix INNER JOIN [01_missingMixData(1)] ON namesmix.code =
    [01_missingMixData(1)].stock
GROUP BY namesmix.Area, namesmix.Gear, [01_missingMixData(1)].tray,
    [01_missingMixData(1)].year, [01_missingMixData(1)].stock
PIVOT [01_missingMixData(1)].Locus;
```

02_base_sum(0)

```
SELECT all_base.stock, all_base.year, all_base.fish
FROM all_base
GROUP BY all_base.stock, all_base.year, all_base.fish;
```

02_base_sum(1)

```
SELECT names.region, names.stock, names_1.stock, names_1.[region name],
    names.[stock name], [02_base_sum(0)].year, Count([02_base_sum(0)].fish) AS
    CountOffish
FROM ([names] INNER JOIN [names] AS names_1 ON names.region = names_1.stock)
    INNER JOIN [02_base_sum(0)] ON names.stock = [02_base_sum(0)].stock
GROUP BY names.region, names.stock, names_1.stock, names_1.[region name],
    names.[stock name], [02_base_sum(0)].year
ORDER BY names.region, names_1.[region name], names.[stock name],
    [02_base_sum(0)].year;
```

02_base_sum(2)

```
SELECT [02_base_sum(1)].region, [02_base_sum(1)].[region name],
    [02_base_sum(1)].[stock name], [02_base_sum(1)].year,
    Max([02_base_sum(1)].CountOffish) AS fish
FROM [02_base_sum(1)]
```

```
GROUP BY [02_base_sum(1)].region, [02_base_sum(1)].[region name],
    [02_base_sum(1)].[stock name], [02_base_sum(1)].year
ORDER BY [02_base_sum(1)].region, [02_base_sum(1)].[stock name],
    [02_base_sum(1)].year;
```

03_fish per locus(1)

```
SELECT all_base.Locus, all_base.stock, all_base.year, all_base.fish
FROM all_base
GROUP BY all_base.Locus, all_base.stock, all_base.year, all_base.fish;
```

03_fish per locus(2)

```
SELECT [03_fish per locus(1)].stock, [03_fish per locus(1)].Locus, Count([03_fish per
    locus(1)].fish) AS totfish
FROM [03_fish per locus(1)]
GROUP BY [03_fish per locus(1)].stock, [03_fish per locus(1)].Locus;
```

04_genomix_sp(0)

```
SELECT all_mix2.Locus, all_mix2.gearea, all_mix2.jday, all_mix2.fish,
    manautobin.New_allele, Sum(all_mix2.Value) AS [value], lociprams.HW_corr
    AS hw_correction
FROM lociprams INNER JOIN (all_mix2 INNER JOIN manautobin ON
    (all_mix2.Locus = manautobin.locus) AND (all_mix2.Allele =
    manautobin.Allele)) ON lociprams.locus = all_mix2.Locus
GROUP BY all_mix2.Locus, all_mix2.gearea, all_mix2.jday, all_mix2.fish,
    manautobin.New_allele, lociprams.HW_corr
HAVING (((Sum(all_mix2.Value))<>0))
ORDER BY all_mix2.Locus, all_mix2.jday, all_mix2.fish, manautobin.New_allele;
```

04_genomix_sp(2)

```
SELECT [04_genomix_sp(0)].Locus, [04_genomix_sp(0)].gearea,
    [04_genomix_sp(0)].Jday, [04_genomix_sp(0)].fish, IIf([hw_correction]="no"
    And [value]=1,[new_allele],Null) AS het, IIf([hw_correction]="no" And
    [value]=2,[new_allele],Null) AS homo,
    IIf([hw_correction]="yes",[New_allele],Null) AS allele,
    IIf([hw_correction]="yes",[value],Null) AS Cvalue,
    [04_genomix_sp(0)].hw_correction
FROM [04_genomix_sp(0)];
```

04_genomix_sp(3)

```
SELECT [04_genomix_sp(2)].Locus, [04_genomix_sp(2)].gearea,
    [04_genomix_sp(2)].Jday, [04_genomix_sp(2)].fish, IIf([het]<10,"0" &
    [het],[het]) AS hetfix, IIf([homo]<10,"0" & [homo],[homo]) AS homfix,
    [04_genomix_sp(2)].allele, [04_genomix_sp(2)].Cvalue,
    [04_genomix_sp(2)].hw_correction
FROM [04_genomix_sp(2)];
```

04_genomix_sp(4)

```
SELECT [04_genomix_sp(3)].Locus, [04_genomix_sp(3)].gearea,
    [04_genomix_sp(3)].jday, [04_genomix_sp(3)].fish,
    First([04_genomix_sp(3)].hetfix) AS het1, Last([04_genomix_sp(3)].hetfix) AS
    het2, [04_genomix_sp(3)].homfix AS hom, [04_genomix_sp(3)].allele,
    [04_genomix_sp(3)].Cvalue
```

```
FROM [04_genomix_sp(3)]
GROUP BY [04_genomix_sp(3)].Locus, [04_genomix_sp(3)].gearea,
    [04_genomix_sp(3)].jday, [04_genomix_sp(3)].fish, [04_genomix_sp(3)].homfix,
    [04_genomix_sp(3)].allele, [04_genomix_sp(3)].Cvalue;
```

04_genomix_sp(5)

```
SELECT [04_genomix_sp(4)].gearea, [04_genomix_sp(4)].jday,
    [04_genomix_sp(4)].fish, [04_genomix_sp(4)].Locus, IIf([het1],[het1] &
    [het2],[hom] & [hom]) AS gen, [04_genomix_sp(4)].allele,
    [04_genomix_sp(4)].Cvalue AS [value]
FROM [04_genomix_sp(4)]
ORDER BY [04_genomix_sp(4)].gearea, [04_genomix_sp(4)].jday,
    [04_genomix_sp(4)].fish, [04_genomix_sp(4)].Locus, IIf([het1],[het1] &
    [het2],[hom] & [hom]), [04_genomix_sp(4)].allele;
```

04_geno_nn(1)

```
SELECT all_base.stock, all_base.year, all_base.fish, all_base.Locus,
    manautobin.New_allele, all_base.Value AS [value], lociprams.bin_allele AS
    max_allele
FROM (all_base INNER JOIN manautobin ON (all_base.Locus = manautobin.locus)
    AND (all_base.Allele = manautobin.Allele)) INNER JOIN lociprams ON
    manautobin.locus = lociprams.locus
WHERE (((all_base.Value)<>0));
```

04_geno_nn(2)

```
SELECT [04_geno_nn(1)].Locus, [04_geno_nn(1)].stock, [04_geno_nn(1)].year,
    [04_geno_nn(1)].fish, IIf([value]=1,[new_allele],Null) AS het,
    IIf([value]=2,[new_allele],Null) AS homo, [04_geno_nn(1)].value AS Cvalue,
    [04_geno_nn(1)].max_allele
FROM [04_geno_nn(1)];
```

04_geno_nn(3)

```
SELECT [04_geno_nn(2)].Locus, [04_geno_nn(2)].stock, [04_geno_nn(2)].year,
    [04_geno_nn(2)].fish, IIf([het]<10,"0" & [het],[het]) AS hetfix,
    IIf([homo]<10,"0" & [homo],[homo]) AS homfix, [04_geno_nn(2)].Cvalue,
    [04_geno_nn(2)].max_allele
FROM [04_geno_nn(2)];
```

04_geno_nn(4)

```
SELECT [04_geno_nn(3)].Locus, [04_geno_nn(3)].stock, [04_geno_nn(3)].year,
    [04_geno_nn(3)].fish, Min([04_geno_nn(3)].hetfix) AS het1,
    Max([04_geno_nn(3)].hetfix) AS het2, [04_geno_nn(3)].homfix AS hom,
    [04_geno_nn(3)].max_allele
FROM [04_geno_nn(3)]
GROUP BY [04_geno_nn(3)].Locus, [04_geno_nn(3)].stock, [04_geno_nn(3)].year,
    [04_geno_nn(3)].fish, [04_geno_nn(3)].homfix, [04_geno_nn(3)].max_allele;
```

04_geno_nn(5)

```
SELECT [04_geno_nn(4)].stock, [04_geno_nn(4)].year, [04_geno_nn(4)].fish,
    [04_geno_nn(4)].Locus, IIf([het1],[het1] & [het2],[hom] & [hom]) AS gen,
    [04_geno_nn(4)].max_allele
FROM [04_geno_nn(4)];
```

04_geno_nn(6)

```
SELECT [04_geno_nn(5)].stock, [04_geno_nn(5)].year, [04_geno_nn(5)].fish,
[04_geno_nn(5)].Locus, [04_geno_nn(5)].gen, [04_geno_nn(5)].max_allele
FROM [04_geno_nn(5)]
ORDER BY [04_geno_nn(5)].stock, [04_geno_nn(5)].year, [04_geno_nn(5)].fish,
    [04_geno_nn(5)].Locus;
```

04_geno_sp(1)

```
SELECT all_base.stock, all_base.year, all_base.fish, all_base.Locus,
    manautobin.New_allele, all_base.Value AS [value], lociprams.HW_corr AS
    Hw_correction
FROM (all_base INNER JOIN manautobin ON (all_base.Locus = manautobin.locus)
    AND (all_base.Allele = manautobin.Allele)) INNER JOIN lociprams ON
    manautobin.locus = lociprams.locus
WHERE (((all_base.Value)<>0));
```

04_geno_sp(2)

```
SELECT [04_geno_sp(1)].Locus, [04_geno_sp(1)].stock, [04_geno_sp(1)].year,
    [04_geno_sp(1)].fish, IIf([hw_correction]="no" And [value]=1,[new_allele],Null)
    AS het, IIf([hw_correction]="no" And [value]=2,[new_allele],Null) AS homo,
    IIf([hw_correction]="yes",[New_allele],Null) AS n_allele,
    IIf([hw_correction]="yes",[value],Null) AS Cvalue,
    [04_geno_sp(1)].Hw_correction
FROM [04_geno_sp(1)];
```

04_geno_sp(3)

```
SELECT [04_geno_sp(2)].Locus, [04_geno_sp(2)].stock, [04_geno_sp(2)].year,
    [04_geno_sp(2)].fish, IIf([het]<10,"0" & [het],[het]) AS hetfix, IIf([homo]<10,"0"
    & [homo],[homo]) AS homfix, [04_geno_sp(2)].n_allele, [04_geno_sp(2)].Cvalue
FROM [04_geno_sp(2)];
```

04_geno_sp(4)

```
SELECT [04_geno_sp(3)].Locus, [04_geno_sp(3)].stock, [04_geno_sp(3)].year,
    [04_geno_sp(3)].fish, Min([04_geno_sp(3)].hetfix) AS het1,
    Max([04_geno_sp(3)].hetfix) AS het2, [04_geno_sp(3)].homfix AS hom,
    [04_geno_sp(3)].n_allele, Sum([04_geno_sp(3)].Cvalue) AS [value]
FROM [04_geno_sp(3)]
GROUP BY [04_geno_sp(3)].Locus, [04_geno_sp(3)].stock, [04_geno_sp(3)].year,
    [04_geno_sp(3)].fish, [04_geno_sp(3)].homfix, [04_geno_sp(3)].n_allele;
```

04_geno_sp(5)

```
SELECT [04_geno_sp(4)].stock, [04_geno_sp(4)].year, [04_geno_sp(4)].fish,
    [04_geno_sp(4)].Locus, IIf([het1],[het1] & [het2],[hom] & [hom]) AS gen,
    [04_geno_sp(4)].n_allele, [04_geno_sp(4)].value
FROM [04_geno_sp(4)];
```

04_geno_sp(6)

```
SELECT NAMES.[stock name], [04_geno_sp(5)].stock, [04_geno_sp(5)].Locus,
    [04_geno_sp(5)].gen AS geno, [04_geno_sp(5)].n_allele AS allele,
    Sum([04_geno_sp(5)].value) AS alle_cnt, Count([04_geno_sp(5)].gen) AS
    geno_cnt, lociprams.bin_allele AS max_allele, lociprams.HW_corr AS
    hw_correction
```

```
FROM lociprams INNER JOIN ([NAMES] INNER JOIN [04_geno_sp(5)] ON
    NAMES.stock = [04_geno_sp(5)].stock) ON lociprams.locus =
    [04_geno_sp(5)].Locus
GROUP BY NAMES.[stock name], [04_geno_sp(5)].stock, [04_geno_sp(5)].Locus,
    [04_geno_sp(5)].gen, [04_geno_sp(5)].n_allele, lociprams.bin_allele,
    lociprams.HW_corr
ORDER BY [04_geno_sp(5)].stock, [04_geno_sp(5)].Locus, [04_geno_sp(5)].gen,
    [04_geno_sp(5)].n_allele;
```

04_geno_sp(7)

```
SELECT [04_geno_sp(6)].[stock name], [04_geno_sp(6)].stock, [04_geno_sp(6)].Locus,
    [04_geno_sp(6)].geno, [04_geno_sp(6)].allele, [04_geno_sp(6)].geno_cnt AS
    cntgeno, [03_fish per locus(2)].totfish AS xfish, [alle_cnt]/(2*[totfish]) AS freq,
    [04_geno_sp(6)].max_allele AS maxallele, [04_geno_sp(6)].hw_correction
FROM [04_geno_sp(6)] INNER JOIN [03_fish per locus(2)] ON ([04_geno_sp(6)].stock
    = [03_fish per locus(2)].stock) AND ([04_geno_sp(6)].Locus = [03_fish per
    locus(2)].Locus)
ORDER BY [04_geno_sp(6)].stock, [04_geno_sp(6)].Locus, [04_geno_sp(6)].geno,
    [04_geno_sp(6)].allele;
```

04_julian day select(1)

```
SELECT all_mix.year AS jday
FROM all_mix
GROUP BY all_mix.year;
```

04_julian day select(2)

```
SELECT all_mix.stock AS gearea, all_mix.year AS jday, all_mix.fish, all_mix.Locus,
    all_mix.Allele, all_mix.Value INTO all_mix2
FROM all_mix
WHERE (((all_mix.year)>=[forms]![select julian day]![julianmin] And
    (all_mix.year)<=[forms]![select julian day]![julianmax]));
```

05_gda header(1)

```
SELECT all_base.stock, all_base.Locus, lociprams.allele
FROM all_base INNER JOIN lociprams ON all_base.Locus = lociprams.locus
GROUP BY all_base.stock, all_base.Locus, lociprams.allele;
```

05_gda header(2)

```
SELECT Count([05_gda header(1)].stock) AS CountOfstock, [05_gda header(1)].Locus,
    [05_gda header(1)].allele AS max_allele
FROM [05_gda header(1)]
GROUP BY [05_gda header(1)].Locus, [05_gda header(1)].allele;
```

05_gda header(3)

```
SELECT Count([05_gda header(2)].Locus) AS locus, Max([05_gda
    header(2)].CountOfstock) AS Stock
FROM [05_gda header(2)];
```

05_gdamix header(1)

```
SELECT all_mix2.gearea AS stock, all_mix2.Locus, lociprams.allele
FROM all_mix2 INNER JOIN lociprams ON all_mix2.Locus = lociprams.locus
GROUP BY all_mix2.gearea, all_mix2.Locus, lociprams.allele;
```

05_gdamix header(2)

```
SELECT [05_gdamix header(1)].Locus, Count([05_gdamix header(1)].stock) AS
    stockcnt, lociprams.allele AS allelecnt
FROM [05_gdamix header(1)] INNER JOIN lociprams ON [05_gdamix
    header(1)].Locus = lociprams.locus
GROUP BY [05_gdamix header(1)].Locus, lociprams.allele;
```

05_gdamix header(3)

```
SELECT Count([05_gdamix header(2)].Locus) AS locus, [05_gdamix
    header(2)].stockcnt AS stock
FROM [05_gdamix header(2)]
GROUP BY [05_gdamix header(2)].stockcnt;
```

05_phylip header(1)

```
SELECT [05_phylip(1)].Locus, [05_phylip(1)].stock
FROM [05_phylip(1)]
GROUP BY [05_phylip(1)].Locus, [05_phylip(1)].stock;
```

05_phylip header(2)

```
SELECT [05_phylip header(1)].Locus, Count([05_phylip header(1)].stock) AS stockcnt,
    lociprams.allele AS allelecnt
FROM [05_phylip header(1)] INNER JOIN lociprams ON [05_phylip header(1)].Locus
    = lociprams.locus
GROUP BY [05_phylip header(1)].Locus, lociprams.allele;
```

05_phylip(1)

```
SELECT all_base.stock, all_base.Locus, all_base.Allele, Sum(all_base.Value) AS
    Allelesum
FROM all_base INNER JOIN lociprams ON all_base.Locus = lociprams.locus
GROUP BY all_base.stock, all_base.Locus, all_base.Allele;
```

05_phylip(2)

```
SELECT all_base.stock, all_base.Locus, Sum(all_base.Value) AS locusum
FROM lociprams INNER JOIN all_base ON lociprams.locus = all_base.Locus
GROUP BY all_base.stock, all_base.Locus;
```

05_phylip(3)

```
SELECT NAMES.[stock name], [05_phylip(1)].Locus, [05_phylip(1)].Allele,
    [allelesum]/[locusum] AS [value], lociprams.allele AS maxunbin
FROM ([NAMES] INNER JOIN ([05_phylip(1)] INNER JOIN [05_phylip(2)] ON
    ([05_phylip(1)].Locus = [05_phylip(2)].Locus) AND ([05_phylip(1)].stock =
    [05_phylip(2)].stock)) ON NAMES.stock = [05_phylip(1)].stock) INNER JOIN
    lociprams ON [05_phylip(2)].Locus = lociprams.locus
ORDER BY NAMES.[stock name], [05_phylip(1)].Locus, [05_phylip(1)].Allele;
```

06_check regions(1)

```
SELECT NAMES.stock AS [region code], NAMES.[region name]
FROM [NAMES]
WHERE (((NAMES.[region name]) Is Not Null));
```

06_check regions(2)

```
SELECT [06_check regions(1)].[region name] AS regname, [06_check
    regions(1)].[region code] AS regcode, NAMES.[stock name] AS stock,
    NAMES.stock AS stkcode
```

```
FROM [06_check regions(1)] INNER JOIN (all_base INNER JOIN [NAMES] ON
        all_base.stock = NAMES.stock) ON [06_check regions(1)].[region code] =
        NAMES.region
GROUP BY [06_check regions(1)].[region name], [06_check regions(1)].[region code],
        NAMES.[stock name], NAMES.stock
ORDER BY [06_check regions(1)].[region code];
```

06_check regions(3)

```
SELECT Count([06_check regions(2)].stock) AS stockcnt
FROM [06_check regions(2)];
```

07_gda(1)

```
SELECT all_base.stock, all_base.year, all_base.fish, all_base.Locus,
        IIf([all_base].[Allele]<10,0 & [all_base].[allele],[all_base].[allele]) AS alle,
        all_base.Value, lociprams.HW_corr AS [H_W correction], lociprams.allele
FROM all_base INNER JOIN lociprams ON all_base.Locus = lociprams.locus
WHERE (((all_base.Value)<>0) AND ((lociprams.HW_corr)="yes"));
```

07_gda(1)b

```
SELECT [07_gda(1)a].stock, [07_gda(1)a].year, [07_gda(1)a].fish,
        Count([07_gda(1)a].Locus) AS cntlocus
FROM [07_gda(1)a]
GROUP BY [07_gda(1)a].stock, [07_gda(1)a].year, [07_gda(1)a].fish;
```

07_gda(1)c

```
SELECT [07_gda(1)b].stock, [07_gda(1)b].year, [07_gda(1)b].fish,
        [07_gda(1)b].cntlocus
FROM [07_gda(1)b];
```

07_gda(2)

```
SELECT [07_gda(1)].stock, [07_gda(1)].year, [07_gda(1)].fish, [07_gda(1)].Locus,
        Min(IIf([value]>0,[alle],Null)) AS A, Max(IIf([value]>0,[alle])) AS B
FROM [07_gda(1)] INNER JOIN [07_gda(1)c] ON ([07_gda(1)].fish =
        [07_gda(1)c].fish) AND ([07_gda(1)].year = [07_gda(1)c].year) AND
        ([07_gda(1)].stock = [07_gda(1)c].stock)
GROUP BY [07_gda(1)].stock, [07_gda(1)].year, [07_gda(1)].fish, [07_gda(1)].Locus;
```

07_gda(2a)

```
SELECT [07_gda(2)].stock, [07_gda(2)].year, [07_gda(2)].fish,
        Count([07_gda(2)].Locus) AS LociCnt
FROM [07_gda(2)]
GROUP BY [07_gda(2)].stock, [07_gda(2)].year, [07_gda(2)].fish;
```

07_gda(2b)

```
SELECT [07_gda(2)].stock, [07_gda(2)].year, [07_gda(2)].fish, [07_gda(2)].Locus,
        [07_gda(2)].A, [07_gda(2)].B, [07_gda(2a)].LociCnt
FROM [07_gda(2a)] INNER JOIN [07_gda(2)] ON ([07_gda(2a)].fish =
        [07_gda(2)].fish) AND ([07_gda(2a)].year = [07_gda(2)].year) AND
        ([07_gda(2a)].stock = [07_gda(2)].stock);
```

07_gda(3)

```
SELECT NAMES.[stock name], [07_gda(2)].year, [07_gda(2)].fish, [07_gda(2)].Locus,
        [A] & "/" & [B] AS geno
FROM [NAMES] INNER JOIN [07_gda(2)] ON NAMES.stock = [07_gda(2)].stock
```

```
ORDER BY NAMES.[stock name], [07_gda(2)].year, [07_gda(2)].fish,
[07_gda(2)].Locus;
```

07_geneclassmix(1)

```
SELECT all_mix2.gearea AS stock, all_mix2.jday AS [year], all_mix2.fish,
all_mix2.Locus, IIf([all_mix2].[Allele]<10,0 & [all_mix2].[allele],[all_mix2].[allele]) AS
alle, all_mix2.Value, lociprams.HW_corr AS [H_W correction], lociprams.allele AS
max_allele
FROM all_mix2 INNER JOIN lociprams ON all_mix2.Locus = lociprams.locus
WHERE (((all_mix2.Value)<>0) AND ((lociprams.HW_corr)="yes"));
```

07_geneclassmix(1)a

```
SELECT [07_geneclassmix(1)].stock, [07_geneclassmix(1)].year,
[07_geneclassmix(1)].fish, [07_geneclassmix(1)].Locus,
[07_geneclassmix(1)].max_allele
FROM [07_geneclassmix(1)]
GROUP BY [07_geneclassmix(1)].stock, [07_geneclassmix(1)].year,
[07_geneclassmix(1)].fish, [07_geneclassmix(1)].Locus,
[07_geneclassmix(1)].max_allele;
```

07_geneclassmix(1)b

```
SELECT [07_geneclassmix(1)a].stock, [07_geneclassmix(1)a].year,
[07_geneclassmix(1)a].fish, Count([07_geneclassmix(1)a].Locus) AS cntlocus
FROM [07_geneclassmix(1)a]
GROUP BY [07_geneclassmix(1)a].stock, [07_geneclassmix(1)a].year,
[07_geneclassmix(1)a].fish;
```

07_geneclassmix(1)c

```
SELECT [07_geneclassmix(1)b].stock, [07_geneclassmix(1)b].year,
[07_geneclassmix(1)b].fish, [07_geneclassmix(1)b].cntlocus
FROM [07_geneclassmix(1)b];
```

07_geneclassmix(2)

```
SELECT [07_geneclassmix(1)].stock, [07_geneclassmix(1)].year,
[07_geneclassmix(1)].fish, [07_geneclassmix(1)].Locus, Min(IIf([value]>0,[alle],Null))
AS A, Max(IIf([value]>0,[alle])) AS B, [07_geneclassmix(1)c].cntlocus
FROM [07_geneclassmix(1)] INNER JOIN [07_geneclassmix(1)c] ON
([07_geneclassmix(1)].fish = [07_geneclassmix(1)c].fish) AND
([07_geneclassmix(1)].year = [07_geneclassmix(1)c].year) AND
([07_geneclassmix(1)].stock = [07_geneclassmix(1)c].stock)
GROUP BY [07_geneclassmix(1)].stock, [07_geneclassmix(1)].year,
[07_geneclassmix(1)].fish, [07_geneclassmix(1)].Locus, [07_geneclassmix(1)c].cntlocus;
```

07_geneclassmix(3)

```
SELECT namesmix.Area AS [stock name], [07_geneclassmix(2)].year,
[07_geneclassmix(2)].fish, [07_geneclassmix(2)].Locus, [A] & [B] AS geno,
[07_geneclassmix(2)].cntlocus AS locicnt
FROM [07_geneclassmix(2)] INNER JOIN namesmix ON [07_geneclassmix(2)].stock =
namesmix.code
ORDER BY namesmix.Area, [07_geneclassmix(2)].year, [07_geneclassmix(2)].fish,
[07_geneclassmix(2)].Locus;
```

07_genepop(1)

```
SELECT NAMES.[stock name], [07_gda(2b)].year, [07_gda(2b)].fish,
[07_gda(2b)].Locus, [A] & [B] AS geno, [07_gda(2b)].LociCnt
FROM [NAMES] INNER JOIN [07_gda(2b)] ON NAMES.stock = [07_gda(2b)].stock
ORDER BY NAMES.[stock name], [07_gda(2b)].year, [07_gda(2b)].fish,
[07_gda(2b)].Locus;
```

09_binfreq(1)

```
SELECT all_base.stock, all_base.year, all_base.fish, all_base.Locus, all_base.Allele,
all_base.Value
FROM all_base;
```

09_binfreq(2)

```
SELECT [09_binfreq(1)].Locus, [09_binfreq(1)].Allele, Sum([09_binfreq(1)].Value) AS
SumOfValue
FROM [09_binfreq(1)]
GROUP BY [09_binfreq(1)].Locus, [09_binfreq(1)].Allele;
```

09_binfreq(3)

```
SELECT [09_binfreq(2)].Locus, Sum([09_binfreq(2)].SumOfValue) AS
SumOfSumOfValue
FROM [09_binfreq(2)]
GROUP BY [09_binfreq(2)].Locus;
```

09_binfreq(4)

```
SELECT [09_binfreq(2)].Locus, [09_binfreq(2)].Allele,
[sumofvalue]/[SumOfSumOfValue] AS freq
FROM [09_binfreq(2)] INNER JOIN [09_binfreq(3)] ON [09_binfreq(2)].Locus =
[09_binfreq(3)].Locus
ORDER BY [09_binfreq(2)].Locus, [09_binfreq(2)].Allele;
```

10_bayesbase(1)

```
SELECT all_base.stock, lociprams.locus_code, all_base.Allele, Sum(all_base.Value) AS
AlleCnt, lociprams.allele AS max_allele
FROM (lociprams INNER JOIN all_base ON lociprams.locus = all_base.Locus) INNER
JOIN [NAMES] ON all_base.stock = NAMES.stock
GROUP BY all_base.stock, lociprams.locus_code, all_base.Allele, lociprams.allele;
```

10_bayesbase(2)

```
SELECT all_base.stock, lociprams.locus_code, Sum(all_base.Value) AS AlleTot,
lociprams.bin_allele AS max_allele
FROM (lociprams INNER JOIN all_base ON lociprams.locus = all_base.Locus) INNER
JOIN [NAMES] ON all_base.stock = NAMES.stock
GROUP BY all_base.stock, lociprams.locus_code, lociprams.bin_allele;
```

10_bayesbase(3)

```
SELECT [10_bayesbase(2)].stock, [10_bayesbase(2)].locus_code,
[10_bayesbase(1)].Allele, [alleCnt]/[AlleTot] AS freq, [10_bayesbase(1)].max_allele,
[10_bayesbase(2)].AlleTot
FROM [10_bayesbase(2)] INNER JOIN [10_bayesbase(1)] ON
([10_bayesbase(2)].locus_code = [10_bayesbase(1)].locus_code) AND
([10_bayesbase(2)].stock = [10_bayesbase(1)].stock);
```

11_allbase_sort

```
SELECT all_base.Locus
FROM all_base
ORDER BY all_base.Locus;
```

11_ImportSpecTbl_sort

```
SELECT ImportSpecTbl.bsemix, ImportSpecTbl.species, ImportSpecTbl.locus,
ImportSpecTbl.importSpecs, ImportSpecTbl.importFile, ImportSpecTbl.alleles
FROM ImportSpecTbl INNER JOIN temp ON ImportSpecTbl.species = temp.species
WHERE (((ImportSpecTbl.bsemix)="base"))
ORDER BY ImportSpecTbl.bsemix, ImportSpecTbl.species, ImportSpecTbl.locus;
```

11_manautobin_allele_cnts

```
SELECT manautobin.locus, Max(manautobin.Allele) AS allele,
Max(manautobin.New_allele) AS Bin_allele
FROM manautobin
GROUP BY manautobin.locus;
```

12_ListImportSpecs(1)

```
SELECT ImportSpecTbl.importSpecs INTO TempImportSpecs
FROM ImportSpecTbl
WHERE
(((ImportSpecTbl.importSpecs)=[forms]![ListImportSpec]![ComboListImportSpecs]));
```

13_manautobincounts(1)

```
SELECT manautobin.locus, Count(manautobin.New_allele) AS CntAll
FROM manautobin
GROUP BY manautobin.locus;
```

13_manautobincounts(2)

```
SELECT manautobin.locus, manautobin.New_allele, Count(manautobin.New_allele) AS
cntnew_allele
FROM manautobin
GROUP BY manautobin.locus, manautobin.New_allele;
```

13_manautobincounts(3)

```
SELECT [13_manautobincounts(2)].locus, [13_manautobincounts(2)].new_allele,
[cntnew_allele]/[CntAll] AS freq
FROM [13_manautobincounts(1)] INNER JOIN [13_manautobincounts(2)] ON
[13_manautobincounts(1)].locus = [13_manautobincounts(2)].locus;
```

14_bsescreen<>2allele(1)

```
SELECT all_base.Locus, all_base.stock, all_base.year, all_base.fish,
Sum(all_base.Value) AS SumOfValue
FROM all_base
GROUP BY all_base.Locus, all_base.stock, all_base.year, all_base.fish
HAVING (((Sum(all_base.Value))<>2))
ORDER BY all_base.Locus, all_base.stock, all_base.year, all_base.fish;
```

14_mixscreen<>2allele(1)

```
SELECT all_mix2.gearea, all_mix2.jday, all_mix2.fish, all_mix2.Locus,
Sum(all_mix2.Value) AS SumOfValue
FROM all_mix2
GROUP BY all_mix2.gearea, all_mix2.jday, all_mix2.fish, all_mix2.Locus
HAVING (((Sum(all_mix2.Value))<>2))
```

```
ORDER BY all_mix2.gearea, all_mix2.jday, all_mix2.fish;
```
14_showbse(1)
```
SELECT all_base.tray, all_base.Locus, all_base.stock, all_base.year, all_base.fish
FROM all_base
GROUP BY all_base.tray, all_base.Locus, all_base.stock, all_base.year, all_base.fish;
```
14_showbse(2)
```
TRANSFORM Count([14_showbse(1)].fish) AS cntfish
SELECT names.region, names.[stock name], [14_showbse(1)].stock
FROM [14_showbse(1)] INNER JOIN [names] ON [14_showbse(1)].stock =
names.stock
GROUP BY names.region, names.[stock name], [14_showbse(1)].stock
ORDER BY names.region, names.[stock name], [14_showbse(1)].stock
PIVOT [14_showbse(1)].Locus;
```
14_showmix(1)
```
SELECT all_mix2.gearea, all_mix2.Locus, all_mix2.fish, all_mix2.jday
FROM all_mix2
GROUP BY all_mix2.gearea, all_mix2.Locus, all_mix2.fish, all_mix2.jday;
```
14_showmix(2)
```
TRANSFORM Count([14_showmix(1)].fish) AS cntfish
SELECT namesMix.Area, namesMix.Gear, [14_showmix(1)].gearea AS AGcode,
Min([14_showmix(1)].Jday) AS minJday, Max([14_showmix(1)].Jday) AS maxjday
FROM namesMix INNER JOIN [14_showmix(1)] ON namesMix.code =
[14_showmix(1)].gearea
GROUP BY namesMix.Area, namesMix.Gear, [14_showmix(1)].gearea
ORDER BY [14_showmix(1)].gearea
PIVOT [14_showmix(1)].Locus;
```
Add all baseline
```
INSERT INTO main_aggregation ( stock, species, [special name] )
SELECT names.stock, [Forms]![Add to baseline]![species] AS species, [Forms]![Add to
baseline]![agsel] AS [special name]
FROM [names];
```
Add all mixture
```
INSERT INTO [main_aggregate mix] ( code, species, [special namemix] )
SELECT namesmix.code, [forms]![Add to mixture]![species] AS species, [forms]![Add
to mixture]![specialnamemixcombo] AS [specialname mix]
FROM namesmix;
```
basescreen(1)
```
SELECT all_base.stock, all_base.year, all_base.fish, all_base.Locus,
Sum(all_base.Value) AS [value]
FROM all_base
GROUP BY all_base.stock, all_base.year, all_base.fish, all_base.Locus
ORDER BY all_base.stock, all_base.year;
```
basescreen(2)
```
SELECT [basescreen(1)].Locus, [basescreen(1)].stock, [basescreen(1)].year,
[basescreen(1)].fish, [basescreen(1)].value
FROM [basescreen(1)]
```

WHERE (((IIf([value]<>2 Or [value] Is Null,"out","in"))="in"));

basescreen(3)-removed fish

SELECT [basescreen(1)].Locus, [basescreen(1)].year, [basescreen(1)].stock, [basescreen(1)].fish, [basescreen(1)].value
FROM [basescreen(1)]
WHERE (((IIf([value]<>2 Or [value] Is Null,"out","in"))="out"))
ORDER BY [basescreen(1)].Locus, [basescreen(1)].year, [basescreen(1)].stock, [basescreen(1)].fish;

Del Aggregate

DELETE main_aggregation.*, main_aggregation.[special name], main_aggregation.stock
FROM main_aggregation
WHERE (((main_aggregation.[special name])=[Forms]![Add to baseline]![agsel]) AND ((main_aggregation.stock)=[Forms]![Add to baseline]![List12]));

Del Aggregate all

DELETE main_aggregation.species, main_aggregation.[special name]
FROM main_aggregation
WHERE (((main_aggregation.[special name])=[forms]![add to baseline]![agsel]));

Del Aggregate all mix

DELETE [main_aggregate mix].species, [main_aggregate mix].[special namemix]
FROM [main_aggregate mix]
WHERE ((([main_aggregate mix].[special namemix])=[forms]![add to mixture]![specialnamemixcombo]));

Del Aggregate mix

DELETE [main_aggregate mix].*, [main_aggregate mix].species, [main_aggregate mix].code
FROM [main_aggregate mix]
WHERE ((([main_aggregate mix].species)=[Forms]![Add to mixture]![agselmix]) AND (([main_aggregate mix].code)=[Forms]![Add to mixture]![list13]));

Appendix B. Form Properties

Form : Main Menu
Control: none
ComboBox:"Species" SpeciesCombo
RowSource: SELECT ImportSpecTbl.species FROM ImportSpecTbl GROUP BY ImportSpecTbl.species;
RunMacro: switch species
Vbcode: got focus
Form.Refresh
TxtBox "Baseline Data"
CmdButton "Make Pop Group" Command17
Vbcode: on_click
Dim stDocName As String
Dim stLinkCriteria As String
stDocName = "Add to baseline"
DoCmd.OpenForm stDocName, , , stLinkCriteria
Form : "Add to Baseline"
Control:main_aggregation(Tbl)
Vbcode:Form_open
agsel.Requery
species.Requery
List2.Requery
List12.Requery
DoCmd.GoToRecord , , acNewRec
ComboBox:"Add/Edit Group Name"Agsel
Control:special name
RowSource: SELECT DISTINCTROW main_aggregation.[special name], main_aggregation.species FROM main_aggregation INNER JOIN temp ON main_aggregation.species = temp.species GROUP BY main_aggregation.[special name], main_aggregation.species;
Vbcode: List12.Requery
ComboBox:"Species" Species
Control:species
RowSource: SELECT DISTINCTROW [temp].[species] FROM [temp];
ListBox:List2
Control:stock
RowSource: SELECT DISTINCTROW Names.region, Names.[stock name], Names.stock FROM Names ORDER BY Names.stock;
ListBox:List12
Control:stock

RowSource: SELECT names.[stock name], main_aggregation.stock FROM names INNER JOIN (temp INNER JOIN main_aggregation ON temp.species = main_aggregation.species) ON names.stock = main_aggregation.stock WHERE (((main_aggregation.[special name])=[forms]![Add to baseline]![agsel])) ORDER BY main_aggregation.stock;

CmdButton: ">" Command8

Vbcode:on_click
save = agsel.Value
spp = species.Value
DoCmd.GoToRecord , , acNewRec
agsel.Requery
species.Requery
agsel.Value = save
species.Value = spp
List12.Requery

CmdButton: "<" Command9

Vbcode:on_click
DoCmd.SetWarnings False
DoCmd.OpenQuery ("Del Aggregate")
List12.Requery

CmdButton: ">>" AddAlltoBaseCmdButton

Vbcode: on_click
DoCmd.SetWarnings False
DoCmd.OpenQuery ("Add all baseline")
List12.Requery

CmdButton: "<<" BaseDeleteAllCmdButton

Vbcode: on_click
Dim stDocName As String
DoCmd.SetWarnings 0
stDocName = "Del Aggregate all"
DoCmd.OpenQuery stDocName, acNormal, acEdit
agsel.Requery
List2.Requery
List12.Requery

CmdButton: "Close" CloseFrmBse

Vbcode: on_click
DoCmd.Close acForm, "add to baseline"

ComboBox: "Pop Group:" Special

ControlSource: unbound
RowSource: SELECT DISTINCTROW main_aggregation.[special name] FROM main_aggregation INNER JOIN temp ON main_aggregation.species = temp.species GROUP BY main_aggregation.[special name];

CmdButton: "Select Pops" select stocks

RunMacro: selects stocks.selectbase

CmdButton: "<> 2 alleles" Command65
Vbcode: on_click
Dim stDocName As String
stDocName = "basescreen(3)-removed fish"
DoCmd.OpenQuery stDocName, acNormal, acEdit
CmdButton: "Show Pops"
Vbcode: on_click
Dim stDocName As String
stDocName = "show fishbase(2)"
DoCmd.OpenQuery stDocName, acNormal, acEdit

TxtBox: "Utilities"
CmdButton: "Load Base Data" Import
RunMacro: import.importbase
CmdButton: "Load Mix Data" Command24
RunMacro: import.importmix
CmdButton: "Manual Bin" Command33
Vbcode: on_click
stDocName = "Allele_Ranges"
DoCmd.OpenForm stDocName, , , stLinkCriteria
Form: Allele_Ranges
Control: mainbin(Tbl)
ComboBox:"Species" SpeciesCombo
Control:species
RowSource: SELECT DISTINCTROW ImportSpecTbl.species FROM ImportSpecTbl GROUP BY ImportSpecTbl.species;
Vbcode: on_change
locus.Value = Null
New_allele.Value = Null
Me.Refresh
ComboBox: "Loci"LocusCombo
Control:locus
RowSource: SELECT DISTINCTROW mainbin.locus FROM mainbin WHERE (((mainbin.species)=[Forms]![Allele_Ranges]![SpeciesCombo])) GROUP BY mainbin.locus;
Vbcode: change
Me.Refresh
Vbcode: gotfocus
Me.Refresh
Vbcode: lossfocus
Me.Refresh
SubForm: Mainbin Subform
SourceObject: Mainbin Subform
ChildLink: species, locus
MasterLink:species,locus
RecordSource:mainbin

Order: mainbin!species,mainbin!locus,...
CmdButton: "Close" Close Allele_Ranges Form
Vbcode: on_click
DoCmd.Close
CmdButton: "Autobin" Command73
Vbcode: on_click
DoCmd.OpenForm "Autobin form"
Form: Autobin form
ComboBox:"Min Bin Freq" Combo1
RowSource: value list
CmdButton: "Run Autobin" CmdRunAutoBin
RunMaco: Autobin
CmdButton: "View Frequencies" CmdViewFreqs
VBcode: stDocName = "ChartAutoManBin"
DoCmd.OpenForm stDocName, , , stLinkCriteria
Form: ChartAutoManBin
UnboundObjectFrame: chart1
RowSource: SELECT [13_manautobincounts(3)].new_allele,
[13_manautobincounts(3)].freq FROM [13_manautobincounts(3)]
WHERE
((([13_manautobincounts(3)].locus)=[Forms]![ChartAutoManBin]!
[Combo1])) GROUP BY [13_manautobincounts(3)].new_allele,
[13_manautobincounts(3)].freq ORDER BY
[13_manautobincounts(3)].new_allele;
ComboBox:"Locus" combo1
RowSource: SELECT DISTINCTROW
[13_manautobincounts(3)].locus FROM
[13_manautobincounts(3)] GROUP BY
[13_manautobincounts(3)].locus;
Vbcode: after_update
chart1.Requery
CmdButton: "Close" CmdCloseViewFreq
Vbcode:on_click
DoCmd.Close
CmdButton: "Close" CmdCloseAutobinForm
Vbcode: on_click
DoCmd.Close
CmdButton: "Loci Table" Command62
Vbcode: on_click
Dim stDocName As String
stDocName = "lociprams"
DoCmd.OpenTable stDocName, acNormal, acEdit
CmdButton: "Edit Paths/Files" ImportSpecsMain
Vbcode:on_click
Dim stDocName As String
Dim stLinkCriteria As String

stDocName = "ImportSpecs"
DoCmd.OpenForm stDocName, , , stLinkCriteria
Form:ImportSpecs
Control:ImportSpecsTbl
ComboBox:"Upload species" ComboImport
Control:species
RowSource: SELECT DISTINCTROW ImportSpecTbl.species FROM ImportSpecTbl GROUP BY ImportSpecTbl.species;
ComboBox: "Path" Combo41
Control:path
RowSource: SELECT DISTINCTROW ImportSpecTbl.Path FROM ImportSpecTbl WHERE (((ImportSpecTbl.species)=[Forms]![ImportSpecs]![comboImport])) GROUP BY ImportSpecTbl.Path;
Vbcode: on_gotfocus
Me.Refresh
SubForm: "Files and Paths" ImportSpecsTblSubform
SourceObject:ImportSpecTblSubform
ChildLink:path
MasterLink:combo41
RowSource: ImportSpecTbl
CmdButton: "View Specs" Command9
Vbcode: on_click
Dim stDocName As String
Dim stLinkCriteria As String
stDocName = "EditImportSpecs"
DoCmd.OpenForm stDocName, , , stLinkCriteria

Form: EditImportSpecs
ComboBox:"Specification Name" importFileName1
RowSource: SELECT DISTINCTROW [MSysIMEXSpecs].[SpecName] FROM [MSysIMEXSpecs];
SubForm: "Row Information" Import Spec Parameters
SourceObject: SubFormEditImportSpecs
Child Links: SpecName
Master Links: ImportFileName1
RowSource: SELECT MSysIMEXSpecs.SpecName, MSysIMEXColumns.FieldName, MSysIMEXColumns.Start, MSysIMEXColumns.Width FROM MSysIMEXColumns INNER JOIN MSysIMEXSpecs ON MSysIMEXColumns.SpecID = MSysIMEXSpecs.SpecID ORDER BY MSysIMEXSpecs.SpecName, MSysIMEXColumns.Start;
CmdButton: "Close" CloseImportSpecsSub
Vbcode: on_click
DoCmd.Close

CmdButton: "Edit Export Path" FormExportPath

Vbcode: on_click

stDocName = "ExportSpecs"

DoCmd.OpenForm stDocName, , , stLinkCriteria

Form: ExportSpecs

SubForm: ExportSpecTbl subform

SourceObject: ExportSpecTbl subform

RecordSource: SELECT ExportSpecs.application, ExportSpecs.output_path FROM ExportSpecs;

CmdButton: "Close" CloseExportSpecs

Vbcode: on_click

DoCmd.Close

CmdButton: "Close" CloseImportSpecsMain

Vbcode:on_click

DoCmd.Close

Appendix C. Visual Basic Code in Modules

Function importbse()

```
 Dim directory, prompt, default, Title, pathStr As String
 Dim TmpTbl, SpecTbl, SpecList As Recordset, varReturnVal As Variant
 On Error GoTo ERRORMSG
 DoCmd.Hourglass True
 Set cdb = CurrentDb
Set TmpTbl = cdb.OpenRecordset("temp")
Set SpecTbl = cdb.OpenRecordset("ImportSpecTbl")

'checks temp file for species, uses to get import data from ImportSpecTbl
Do While Not SpecTbl.EOF
      varReturnVal = SysCmd(acSysCmdSetStatus, "Importing Loci=" & SpecTbl!locus)
   If SpecTbl!species = TmpTbl!species And SpecTbl!bsemix = "base" Then
      DoCmd.TransferText acImportFixed, SpecTbl![importSpecs], "bse_" &
SpecTbl!locus, TmpTbl![Path] & SpecTbl![importFile], False
      pathStr = TmpTbl![Path]
   End If
   SpecTbl.MoveNext
Loop

DoCmd.TransferText acImportFixed, "namesimport", "names", pathStr & "names.txt",
False

   varReturnVal = SysCmd(acSysCmdClearStatus)
   MsgBox "Baseline import successful"

 GoTo FINISH   'normal exit
 varReturnVal = SysCmd(acSysCmdClearStatus)
ERRORMSG:
 If (Err = 3011) Then  'Dir function couldn't find directory.
   MsgBox "Could not find file'" & SpecTbl!Path & " " & SpecTbl!importFile, , Title
   Resume Next
 Else
   MsgBox "An error occurred.  Exiting macro.", , Err & Title
   GoTo FINISH
 End If

FINISH:
varReturnVal = SysCmd(acSysCmdClearStatus)
 End Function
```

Function importmix()

```
Dim directory, prompt, default, Title, pathStr As String
Dim TmpTbl, SpecTbl, SpecList As Recordset, varReturnVal As Variant
On Error GoTo ERRORMSG
DoCmd.Hourglass True
Set cdb = CurrentDb
Set TmpTbl = cdb.OpenRecordset("temp")
Set SpecTbl = cdb.OpenRecordset("ImportSpecTbl")
   'cycles through ImportSpec tble to find parameters
Do While Not SpecTbl.EOF
      varReturnVal = SysCmd(acSysCmdSetStatus, "Importing Loci=" & SpecTbl!locus)
   If SpecTbl!species = TmpTbl!species And SpecTbl!bsemix = "mix" Then
      pathStr = TmpTbl![Path]
      DoCmd.TransferText acImportFixed, SpecTbl![importSpecs], "mix_" &
SpecTbl!locus, TmpTbl![Path] & SpecTbl![importFile], False
   End If
   SpecTbl.MoveNext
Loop
DoCmd.TransferText acImportFixed, "namesmiximport", "namesmix", pathStr &
"namesmix.txt", False

 MsgBox "Mixture file import successful"
 GoTo FINISH   'normal exit
ERRORMSG:
 If (Err = 3011) Then  'Dir function couldn't find directory.
   MsgBox "Could not find file mixture data " & TmpTbl!Path & " " & SpecTbl!locus &
" file", , Title
   Resume Next
 Else
   MsgBox "An error occurred.  Exiting macro.", , Err & Title
   GoTo FINISH
 End If

FINISH:
varReturnVal = SysCmd(acSysCmdClearStatus)
DoCmd.Hourglass False
 End Function


Public Function baseSelectAll()
Dim mainBin, tmpfile, namefile As Recordset
Dim frmname, nameStr As String

Set cdb = CurrentDb()
Set namefile = cdb.OpenRecordset("names")
```

```
nameStr = "SELECT [forms]![add to base]![agsel] AS [special name], names.stock INTO
test"
DoCmd.RunSQL nameStr


End Function

Function phylip_yrcomb()
DoCmd.Hourglass True

    'This function creates a fixed-width text file using the phylip table

Dim stkstr As String * 10
Dim cdb As Database, varReturnVal As Variant
Dim rstGeno, rstExp, inSum As Recordset
Dim intfile, i As Integer, numfields As Integer, ValStr, locstr, Path, msg As String
Dim cnt, gen_stock, gen_locus As String, s_len As Integer, in_query As QueryDef
Dim max_allele, AlleleCnt, keepskcnt As Integer

Set cdb = CurrentDb()
Set rstGeno = cdb.OpenRecordset("05_phylip(3)")
Set inSum = cdb.OpenRecordset("05_phylip header(2)")
Set rstExp = cdb.OpenRecordset("ExportSpecs")

intfile = FreeFile ' Get the # of next free file

Do While Not rstExp.EOF
   If rstExp!Application = "phylip" Then
      Path = rstExp!output_path & "phylip.txt"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile

   cnt = 0

'Outputs header stuff

Do While Not inSum.EOF
   cnt = cnt + 1
   locstr = locstr & " " & inSum!AlleleCnt
   keepskcnt = inSum!stockcnt
   inSum.MoveNext
Loop
   Print #intfile, keepskcnt, cnt
```

```
   Print #intfile, locstr

'End of outputs header stuff

' Inital settings
      gen_locus = rstGeno!locus
      gen_stock = rstGeno![stock name]
      max_allele = rstGeno!maxunbin
      cnt = 0

Do While Not rstGeno.EOF
   ValStr = ""
   stkstr = Format(rstGeno![stock name], "##########")
   s_len = Len(stkstr)
   ValStr = stkstr & Space(10 - s_len)
   gen_stock = rstGeno![stock name]

   Do While Not StrComp(rstGeno![stock name], gen_stock)
      Do While StrComp(gen_locus, rstGeno!locus) = 0 ' loop through each locus
         If Not rstGeno.EOF Then
            varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" &
rstGeno![stock name] & " Locus=" & rstGeno!locus)
         End If
         For i = 1 To max_allele
            If i = rstGeno!allele And StrComp(gen_locus, rstGeno!locus) = 0 Then
               cnt = Format(rstGeno!Value, "0.00000000")
               ValStr = ValStr & " " & cnt
               rstGeno.MoveNext
               If rstGeno.EOF Then GoTo dump
            Else
                  cnt = "0.00000000"
                  ValStr = ValStr & " " & cnt
            End If
         Next i
         If rstGeno![stock name] <> gen_stock Then GoTo outloci
      Loop
outloci:
      Print #intfile, ValStr
      ValStr = "         "
      max_allele = rstGeno!maxunbin
      gen_locus = rstGeno!locus
      If rstGeno![stock name] <> gen_stock Then GoTo outloop
   Loop
   Print #intfile,
outloop:
 Loop
```

```
dump:
   Do While i < max_allele   'fill out extra 0's
      ValStr = ValStr & " " & "0.00000"
      i = i + 1
   Loop
   Print #intfile, ValStr

Close intfile
rstGeno.Close

DoCmd.Hourglass False

varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " created"
MsgBox msg
End Function

Public Function phylip_year()

DoCmd.Hourglass True

    'This function creates a fixed-width text file using the phylip table

    'Header Widths
    'Detail widths
     Dim stkstr As String * 10
     Dim cdb As Database, varReturnVal As Variant
     Dim rstGeno, rstExp, inSum As Recordset
     Dim intfile, i As Integer, numfields As Integer, ValStr, locstr, Path, msg As String
     Dim cnt, gen_stock, gen_locus As String, s_len As Integer, in_query As QueryDef
     Dim max_allele, AlleleCnt, keepskcnt As Integer

     'Set some parameters
     Set cdb = CurrentDb()
     Set rstGeno = cdb.OpenRecordset("05_phylip_year(3)")
     Set inSum = cdb.OpenRecordset("05_phylip headeryear(2)")
   Set rstExp = cdb.OpenRecordset("ExportSpecs")
   intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "phylip" Then
      Path = rstExp![output_path] & "phylip_year.txt"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile
   cnt = 0
```

```
'Outputs header stuff

Do While Not inSum.EOF
   cnt = cnt + 1
   locstr = locstr & " " & inSum!AlleleCnt
   inSum.MoveNext
   If Not inSum.EOF Then
      keepskcnt = inSum!stockcnt
   End If
Loop
      Print #intfile, keepskcnt, cnt
      Print #intfile, locstr

' end of outputs header stuff

  ' Inital settings
      gen_locus = rstGeno!locus
      gen_stock = rstGeno![stockyear]
      max_allele = rstGeno!maxunbin
      cnt = 0

Do While Not rstGeno.EOF
   ValStr = ""
   stkstr = Format(rstGeno![stockyear], "##########")
   s_len = Len(stkstr)
   ValStr = stkstr & Space(10 - s_len)
   gen_stock = rstGeno![stockyear]

   Do While Not StrComp(rstGeno![stockyear], gen_stock)

        Do While StrComp(gen_locus, rstGeno!locus) = 0 ' loop through each locus
           If Not rstGeno.EOF Then
              varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" &
rstGeno![stockyear] & " Locus=" & rstGeno!locus)
           End If
           For i = 1 To max_allele
              If i = rstGeno!allele And StrComp(gen_locus, rstGeno!locus) = 0 Then
                 cnt = Format(rstGeno!Value, "0.00000000")
                 ValStr = ValStr & " " & cnt
                 rstGeno.MoveNext
                    If rstGeno.EOF Then GoTo dump
              Else
                 cnt = "0.00000000"
                 ValStr = ValStr & " " & cnt
              End If
```

```
            Next i
        Loop
        Print #intfile, ValStr
        ValStr = "        "
        max_allele = rstGeno!maxunbin
        gen_locus = rstGeno!locus
        If rstGeno![stockyear] <> gen_stock Then GoTo outloop
    Loop
     Print #intfile,
outloop:
 Loop
dump:
    Do While i < max_allele    'fill out extra 0's
        ValStr = ValStr & " " & "0.00000"
        i = i + 1
    Loop
    Print #intfile, ValStr
    Close intfile
    rstGeno.Close

DoCmd.Hourglass False

varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " created"
MsgBox msg
End Function
```

Function genepop()

```
'updated nov29/2000
Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno, rstExp, in2_sum As Recordset
Dim fld As Field, varReturnVal As Variant
Dim intfile
Dim g_len, f_len, y_len As Integer
Dim gen_stock, gen_fish, gen_year As Integer
Dim cnt, ValStr, gen_locus, msg, Path As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries
Set cdb = CurrentDb()
Set rstGeno = cdb.OpenRecordset("07_genepop(1)")
Set in2_sum = cdb.OpenRecordset("05_gda header(2)")
```

```
Set rstExp = cdb.OpenRecordset("ExportSpecs")
intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "genepop" Then
      Path = rstExp![output_path] & "genepop.bse"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile

'Header information
Print #intfile, "genepop input file from MM 2.3"
Do While Not in2_sum.EOF
   Print #intfile, in2_sum!locus
   in2_sum.MoveNext
   cnt = cnt + 1
Loop

'Count the number of fish in each genotype and output to table by stock and locus
in2_sum.MoveFirst
gen_locus = rstGeno!locus
gen_stock = rstGeno![stock name]
gen_fish = rstGeno!Fish

Print #intfile, "POP"
Do While Not StrComp(rstGeno![stock name], gen_stock)

   Do While Not StrComp(gen_fish, rstGeno!Fish)
   g_len = Len(rstGeno![stock name])
   f_len = Len(gen_fish)
   y_len = Len(rstGeno!Year)
   If Not rstGeno.EOF Then
      varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!Fish)
   End If
   ValStr = rstGeno![stock name] & Space(14 - g_len) & rstGeno!Year & Space(5 -
y_len) & gen_fish & Space(4 - f_len) & " , "
      Do While gen_fish = rstGeno!Fish And gen_year = rstGeno!Year
         If rstGeno!locus = in2_sum!locus And Not in2_sum.EOF Then
            cnt = rstGeno!Geno
            rstGeno.MoveNext
         If Not in2_sum.EOF Then
            in2_sum.MoveNext
         End If
         ValStr = ValStr & " " & cnt
      ElseIf rstGeno!locus > in2_sum!locus Then
```

```
        in2_sum.MoveNext
        ValStr = ValStr & " " & "0000"
      ElseIf rstGeno!locus < in2_sum!locus Then
        ValStr = ValStr & " " & "0000"
      End If
      If rstGeno.EOF Then
        GoTo dump
      End If
      Loop
dump:
      Do While Not in2_sum.EOF 'fill in zero for a fish
        ValStr = ValStr & " " & "0000"
        in2_sum.MoveNext
      Loop

      Print #intfile, ValStr
      ValStr = ""
      If Not rstGeno.EOF Then
        gen_fish = rstGeno!Fish
        gen_year = rstGeno!Year
        in2_sum.MoveFirst
      Else
        GoTo outside
      End If
      If gen_stock <> rstGeno![stock name] Then
        GoTo jumpout
      End If
    Loop
jumpout:
    'If Not rstGeno.EOF Then
    '   varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!fish)
    'End If
    gen_stock = rstGeno![stock name]
    Print #intfile, "POP"
Loop
outside:
varReturnVal = SysCmd(acSysCmdClearStatus)
DoCmd.Hourglass False
msg = Path & " created"
MsgBox msg
Close #intfile
End Function

    Public Function spam_est()
```

```
'This file outputs a spam control file

Dim cdb As Database
Dim tbl As TableDef
Dim rststock, rstMax, rstExp As Recordset
Dim fld As Field
Dim intfile, cnt As Integer, numstr As String, numstr_all As String
Dim varstr, Path, msg As String
Dim i, j, k, l As Integer, max_locus As String, slen As Integer, stlen As Integer
Dim formi, locstr, hw_type, typlen, loclen, ilen, alelen As String, formj As String,
newcode As Integer, oldcode As Integer
Dim gen_locus, calc_geno, pathStr As String, gen_geno As Integer, test As String

' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
    Set cdb = CurrentDb()
    Set rststock = cdb.OpenRecordset("06_check regions(2)")
    Set rstMax = cdb.OpenRecordset("lociprams")
    Set rstExp = cdb.OpenRecordset("ExportSpecs")
    intfile = FreeFile      ' Get the # of next free file
Do While Not rstExp.EOF
  If rstExp![Application] = "spam" Then
    Path = rstExp![output_path]
  End If
  rstExp.MoveNext
Loop
  pathStr = Path & "est.ctl"

  Open pathStr For Output As intfile


 Print #intfile, "* Estimation : Microsat Manager generic output"
 Print #intfile,
 Print #intfile, "* options selected for optimization"
 Print #intfile, "  use IRLS algorithm in optimal search    : f"
 Print #intfile, "  print mixture file                      : f"
 Print #intfile, "  print baseline relative frequencies     : f"
 Print #intfile, "  print conditional genotype probabilities : f"
 Print #intfile, "  print conditional stock probabilities   : f"
 Print #intfile, "  compute likelihood confidence intervals : f"
 Print #intfile, "  compute infitesimal jacknife std. dev.  : f"
 Print #intfile, "  resample mixture frequencies            : t"
 Print #intfile, "  resample baseline                       : t"
```

```
Print #intfile,
Print #intfile, "* control parameters"
k = 0
Do While Not rststock.EOF
 k = 1 + k
 rststock.MoveNext
Loop
l = 0
Do While Not rstMax.EOF
 l = 1 + l
 rstMax.MoveNext
Loop

Print #intfile, "  number of stocks in analysis      : " & k
Print #intfile, "  number of characters in analysis  : " & l
Print #intfile, "  maximum number of genotypes       : 900"
Print #intfile, "  maximum number of classes         : 200"
Print #intfile, "  maximum # of iterations           : 1000"
Print #intfile, "  maximum number of missing loci    :  5"
Print #intfile, "  estimate tolerance                : .10e-05"
Print #intfile, "  likelihood tolerance              : 1.0e-10"
Print #intfile, "  genotype tolerance                :  0"
Print #intfile, "  algorithm switch tolerance        :  0.01"
Print #intfile, "  GPA                          :  90"
Print #intfile, "  number of resamplings             :  100"
Print #intfile, "  random seed                  :  -718805"


Print #intfile,
Print #intfile, "* characters used"
Print #intfile, "  [id #]  [# types]  [kind]  [character]"

 i = 0
 rstMax.MoveFirst
 Do While Not rstMax.EOF
    i = i + 1
    alelen = Len(rstMax!bin_allele)
    If rstMax!HW_corr = "yes" Then hw_type = "locus" Else hw_type = "phenotype"
       typlen = Len(hw_type)
       ilen = Len(i)
       loclen = Len(rstMax!locus)
       locstr = Space(10 - ilen) & i & Space(10 - alelen) & rstMax!bin_allele &
Space(15 - typlen) & hw_type & Space(20 - loclen) & rstMax!locus

    Print #intfile, locstr
    locstr = ""
```

```
    rstMax.MoveNext
  Loop

  Print #intfile, "* populations used in analysis"
  Print #intfile, "   [id #]  [population]  [lev1]"
  rststock.MoveFirst
  Do While Not rststock.EOF
  slen = Len(rststock!stock)
  stlen = Len(rststock!stkcode)
  varstr = Space(5 - stlen) & rststock!stkcode & Space(5) & rststock!stock & Space(25 -
slen) & "    :" & "    :" & Space(5) & rststock!regcode
  Print #intfile, varstr
  varstr = ""
  rststock.MoveNext
    Loop

  rststock.MoveFirst
  varstr = ""

  Print #intfile,
  Print #intfile, "*regions"
  Print #intfile, " [level]  [label]  [Region]"
  newcode = rststock!regcode

  Do While Not rststock.EOF
   If newcode = rststock!regcode And oldcode <> newcode Then
     slen = Len(rststock!regcode)
     varstr = Space(4) & "1" & Space(5) & rststock!regcode & Space(10 - slen) &
rststock!regname
      Print #intfile, varstr
   End If
   oldcode = newcode
  rststock.MoveNext
  If Not rststock.EOF Then
    newcode = rststock!regcode
  End If

    varstr = ""
  Loop

  Print #intfile,
  Print #intfile, "* file"
  Print #intfile, "baseline : " & Path & " spam.bse "
  Print #intfile, "mixture  : " & Path & " spam.mix"
  Print #intfile, "output   : " & Path & "output"
  Print #intfile,
```

```
  Print #intfile, "* run"

 ' turn off hourglass
DoCmd.Hourglass False
 msg = Path & " spam.est created"
 MsgBox msg
 'close all tables
 Close #intfile

    End Function

 Function spam_bse()
'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno, rstMax, rstExp, rstHW As Recordset
Dim fld As Field, varReturnVal As Variant
Dim intfile, numfields As Integer, numstr As String, numstr_all As String
Dim ValStr, msg As String, g_len, f_len, cnt_len As Integer
Dim i, j, k As Integer, max_locus As String, gen_stock As Integer
Dim formi, formj, cnt, max_allele, RetVal, Path As String
Dim gen_locus, calc_geno As String, gen_geno As Integer, test As String
Dim gen1_locus, HW_correction, gen_name As String, alle_cnt As Integer
Dim freq As Double, A(99) As Double, Fish As Integer, fishno As String, nam_str As
String
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
     Set cdb = CurrentDb()
     Set rstGeno = cdb.OpenRecordset("04_geno_sp(6)")
     Set rstMax = cdb.OpenRecordset("lociprams")
     Set rstExp = cdb.OpenRecordset("ExportSpecs")
     intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "spam" Then
      Path = rstExp![output_path] & "spam.bse"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile

' Output as either genotype counts or HW corrected frequencies
gen_locus = rstGeno!locus
```

```
gen1_locus = rstGeno!locus
gen_stock = rstGeno!stock
max_allele = rstGeno!max_allele
HW_correction = rstGeno!HW_correction
gen_name = rstGeno![stock name]

nam_str = "#" & " " & Format(gen_stock, "000") & " " & Format(gen_name,
"00000000000")
Print #intfile, nam_str

Do While Not rstGeno.EOF
   If Not rstGeno.EOF Then
      varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " Locus=" & rstGeno![locus])
   End If

   If HW_correction = "no" Then

   'OUTPUT PHENOTYPES
      k = 0
      g_len = Len(gen_locus)
      f_len = Len(fishno)
      ValStr = gen_locus & Space(8 - g_len)
         For i = 1 To max_allele
         For j = i To max_allele
            formi = Format(i, "00")
            formj = Format(j, "00")
            calc_geno = formi & formj

            If Not rstGeno.EOF Then
               If calc_geno = rstGeno!Geno Then
                  cnt = Format(rstGeno!geno_cnt, "##0")
                  cnt_len = Len(cnt)
                  rstGeno.MoveNext
               End If
            End If
            ValStr = ValStr & " " & cnt
            k = k + cnt_len + 1 'previous values plus additional values
            cnt = 0
         Next j
      Next i
      Print #intfile, ValStr
      ValStr = ""
      If k > 500 Then
         MsgBox "loci over 500 characters long"
      End If
```

```
    k = 0
    Else
'OUTPUT HW CORRECTED VALUES
    g_len = Len(gen_locus)
    f_len = Len(fishno)
    ValStr = gen_locus & Space(8 - g_len)
    i = 1
   Do While StrComp(gen_locus, rstGeno!locus) = 0
      If Not rstGeno.EOF Then
        For i = 1 To max_allele
          If Not rstGeno.EOF Then
            If i = rstGeno!allele And rstGeno!locus = gen_locus Then
              ValStr = ValStr & " " & rstGeno!alle_cnt
              rstGeno.MoveNext
            Else
              ValStr = ValStr & " " & "0"  ' add 0's
            End If
          End If
        Next i
      End If
        Print #intfile, ValStr
        ValStr = ""
    If rstGeno.EOF Then GoTo outloop

    Loop
outloop:

  End If
    If Not rstGeno.EOF Then
      If gen_stock <> rstGeno!stock Then
        Print #intfile,
        gen_name = rstGeno![stock name]
        gen_stock = rstGeno!stock
        nam_str = "#" & " " & Format(gen_stock, "000") & " " & Format(gen_name,
"00000000000")
        Print #intfile, nam_str
      End If
    End If

    If Not rstGeno.EOF Then
      gen_locus = rstGeno!locus
      gen_stock = rstGeno!stock
      max_allele = rstGeno!max_allele
      HW_correction = rstGeno!HW_correction
      gen_name = rstGeno![stock name]
    Else
```

```
        GoTo Loopout
      End If
Loop
Loopout:

' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
 msg = Path & " " & " created"
 MsgBox msg, vbOKOnly

 Close #intfile
End Function

    Public Function spam_sim()

    'This file outputs a spam control file

Dim cdb As Database
Dim tbl As TableDef
Dim rststock, rstMax, rstExp As Recordset
Dim fld As Field
Dim intfile, cnt As Integer, numstr As String, numstr_all As String
Dim varstr, Path, msg As String
Dim i, j, k, l As Integer, max_locus As String, slen As Integer, stlen As Integer
Dim formi, locstr, hw_type, typlen, loclen, ilen, alelen As String, formj As String,
newcode As Integer, oldcode As Integer
Dim gen_locus, calc_geno, pathStr As String, gen_geno As Integer, test As String

' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
      Set cdb = CurrentDb()
      Set rststock = cdb.OpenRecordset("06_check regions(2)")
      Set rstMax = cdb.OpenRecordset("lociprams")
      Set rstExp = cdb.OpenRecordset("ExportSpecs")
      intfile = FreeFile      ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "spam" Then
     Path = rstExp![output_path]
   End If
   rstExp.MoveNext
Loop
   pathStr = Path & "sim.ctl"
```

```
Open pathStr For Output As intfile


Print #intfile, "* Simulation : Microsat Manager generic output"
Print #intfile,
Print #intfile, "* options selected for optimization"
Print #intfile, "  use IRLS algorithm in optimal search    : f"
Print #intfile, "  print mixture file                      : f"
Print #intfile, "  print baseline relative frequencies     : f"
Print #intfile, "  print conditional genotype probabilities : f"
Print #intfile, "  print conditional stock probabilities   : f"
Print #intfile, "  compute likelihood confidence intervals : f"
Print #intfile, "  compute infitesimal jacknife std. dev.  : f"
Print #intfile, "  resample mixture frequencies            : t"
Print #intfile, "  resample baseline                       : t"
Print #intfile,
Print #intfile, "* control parameters"
k = 0
Do While Not rststock.EOF
 k = 1 + k
 rststock.MoveNext
Loop
l = 0
Do While Not rstMax.EOF
 l = 1 + l
 rstMax.MoveNext
Loop

Print #intfile, "  number of stocks in analysis      : " & k
Print #intfile, "  number of characters in analysis  : " & l
Print #intfile, "  maximum number of genotypes       : 900"
Print #intfile, "  maximum number of classes         : 200"
Print #intfile, "  maximum # of iterations           : 1000"
Print #intfile, "  maximum number of missing loci    :  5"
Print #intfile, "  estimate tolerance                : .10e-05"
Print #intfile, "  likelihood tolerance              : 1.0e-10"
Print #intfile, "  genotype tolerance                :  0"
Print #intfile, "  algorithm switch tolerance        :  0.01"
Print #intfile, "  GPA                               :  90"
Print #intfile, "  number of resamplings             :  100"
Print #intfile, "  size of simulated mixture sample  :  150"
Print #intfile, "  random seed                       :  -718805"


Print #intfile,
```

```
Print #intfile, "* characters used"
Print #intfile, "  [id #]  [# types]  [kind]  [character]"

i = 0
rstMax.MoveFirst
Do While Not rstMax.EOF
    i = i + 1
    alelen = Len(rstMax!bin_allele)
    If rstMax!HW_corr = "yes" Then hw_type = "locus" Else hw_type = "phenotype"
      typlen = Len(hw_type)
      ilen = Len(i)
      loclen = Len(rstMax!locus)
      locstr = Space(10 - ilen) & i & Space(10 - alelen) & rstMax!bin_allele &
Space(15 - typlen) & hw_type & Space(20 - loclen) & rstMax!locus

    Print #intfile, locstr
    locstr = ""
    rstMax.MoveNext
Loop

Print #intfile, "* populations used in analysis"
Print #intfile, "   [id #]  [population]  [lev1] [Estimate]"
rststock.MoveFirst
Do While Not rststock.EOF
slen = Len(rststock!stock)
stlen = Len(rststock!stkcode)
varstr = Space(5 - stlen) & rststock!stkcode & Space(5) & rststock!stock & Space(25 -
slen) & "    :" & "    :" & Space(5) & rststock!regcode & "     0"
Print #intfile, varstr
varstr = ""
rststock.MoveNext
  Loop

rststock.MoveFirst
varstr = ""

Print #intfile,
Print #intfile, "*regions"
Print #intfile, " [level]  [label]  [Region]"
newcode = rststock!regcode

Do While Not rststock.EOF
 If newcode = rststock!regcode And oldcode <> newcode Then
    slen = Len(rststock!regcode)
    varstr = Space(4) & "1" & Space(5) & rststock!regcode & Space(10 - slen) &
rststock!regname
```

```
    Print #intfile, varstr
  End If
  oldcode = newcode
 rststock.MoveNext
 If Not rststock.EOF Then
  newcode = rststock!regcode
 End If

   varstr = ""
 Loop

 Print #intfile,
 Print #intfile, "* file"
 Print #intfile, "baseline : " & Path & " spam.bse "
 Print #intfile, "mixture  : " & Path & " spam.mix "
 Print #intfile, "output   : " & Path & "output"
 Print #intfile,
 Print #intfile, "* run"

  ' turn off hourglass
DoCmd.Hourglass False
 msg = Path & " spam.sim created"
 MsgBox msg
 'close all tables
 Close #intfile

    End Function


Function get_fields(table_in As String)

'Parameters:    table_in  the name of the table the procedure populates
'               The table must already be appended to the TableDefs

   Dim j As Integer, Tot As Integer   'counters
      Dim in_records As Recordset    'recordset of in table

   Set cdb = CurrentDb

   'assign in_records to recordset of Table_in
   'AbsolutePosition only works with dynasets or snapshots.


Set in_records = cdb.OpenRecordset(table_in, dbOpenDynaset)

   With in_records
```

```
    ' Populate Recordset.
    .MoveLast
    .MoveFirst

    Do While Not .EOF
      Tot = 0
      'loop through fields in table
      For j = 1 To in_records.Fields.count - 1
        Tot = in_records.Fields(j).Value + Tot
      Next j
      .Edit
       For j = 1 To in_records.Fields.count - 1
         in_records.Fields(j).Value = in_records.Fields(j).Value / Tot
      Next j
      .Update
      .MoveNext
    Loop
  .Close
  End With

End Function

Function Append_Tables(In_table As String, out_table As String, Primer As String)

' Declare variables used in this function
Dim tblOut_Table As TableDef
Dim rcdNewRecSet, rcdOldRecSet As Recordset, intMsgRtn As Integer
Dim fldMyField As Field, varReturnVal As Variant
Dim oldtable As TableDef, xnumfields, numfields As Integer
Dim i As Integer, j As Integer

' Initialize the cdb database variable to the current
' database
'Set cdb = DBEngine.Workspaces(0).Databases(0)

'close all tables
On Error GoTo 0
For i = 0 To cdb.TableDefs.count - 1
   DoCmd.Close A_TABLE, cdb(i).Name
Next i

'update any changes
cdb.TableDefs.Refresh

Set rcdNewRecSet = cdb.OpenRecordset(out_table)
```

```
' Turn on hourglass
DoCmd.Hourglass True

Set rcdOldRecSet = cdb.OpenRecordset(In_table)
numfields = rcdOldRecSet.Fields.count   ' number of columns in table

On Error GoTo 0

' initializes status bar counter
j = 0

' initializes the status bar
varReturnVal = SysCmd(SYSCMD_INITMETER, "Adding  " & Primer,
rcdOldRecSet.RecordCount)

' copies the contents of the table removing "999" into Out_Table
While (Not rcdOldRecSet.EOF)
   For i = 4 To (numfields - 1)
      If rcdOldRecSet(i).Value > 0 And rcdOldRecSet(0) <> "999" Then
      varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
      rcdNewRecSet.AddNew
      rcdNewRecSet.Fields(0).Value = rcdOldRecSet.Fields(0).Value
      rcdNewRecSet.Fields(1).Value = rcdOldRecSet.Fields(1).Value
      rcdNewRecSet.Fields(2).Value = rcdOldRecSet.Fields(2).Value
      rcdNewRecSet.Fields(3).Value = rcdOldRecSet.Fields(3).Value
      rcdNewRecSet.Fields(4).Value = Primer
      rcdNewRecSet.Fields(5).Value = rcdOldRecSet.Fields(i).Name
      rcdNewRecSet.Fields(6).Value = rcdOldRecSet(i).Value
      rcdNewRecSet.Update
      End If
   Next i
   rcdOldRecSet.MoveNext
   j = j + 1
   varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
Wend
varReturnVal = SysCmd(acSysCmdClearStatus)
' turn off hourglass
DoCmd.Hourglass False

' reset status bar
varReturnVal = SysCmd(SYSCMD_CLEARSTATUS)


End Function
```

```
Function Append_Tablesyr(In_table As String, out_table As String, Primer As
String)

' Declare variables used in this function
Dim tblOut_Table As TableDef
Dim rcdNewRecSet, rcdOldRecSet As Recordset, intMsgRtn As Integer
Dim fldMyField As Field, varReturnVal As Variant
Dim oldtable As TableDef, xnumfields, numfields As Integer
Dim i As Integer, j As Integer

' Initialize the cdb database variable to the current
' database
'Set cdb = DBEngine.Workspaces(0).Databases(0)

'close all tables
On Error GoTo 0
For i = 0 To cdb.TableDefs.count - 1
   DoCmd.Close A_TABLE, cdb(i).Name
Next i

'update any changes
cdb.TableDefs.Refresh

Set rcdNewRecSet = cdb.OpenRecordset(out_table)

' Turn on hourglass
DoCmd.Hourglass True

Set rcdOldRecSet = cdb.OpenRecordset(In_table)
numfields = rcdOldRecSet.Fields.count   ' number of columns in table

On Error GoTo 0

' initializes status bar counter
j = 0

' initializes the status bar
varReturnVal = SysCmd(SYSCMD_INITMETER, "Adding  " & Primer,
rcdOldRecSet.RecordCount)

' copies the contents of the table removing "999" into Out_Table
While (Not rcdOldRecSet.EOF)
   For i = 5 To (numfields - 1)
      If rcdOldRecSet(i).Value > 0 And rcdOldRecSet(0) <> "999" Then
      varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
      rcdNewRecSet.AddNew
```

```
        rcdNewRecSet.Fields(0).Value = rcdOldRecSet.Fields(0).Value
        rcdNewRecSet.Fields(1).Value = rcdOldRecSet.Fields(1).Value
        rcdNewRecSet.Fields(2).Value = rcdOldRecSet.Fields(2).Value
        rcdNewRecSet.Fields(3).Value = rcdOldRecSet.Fields(3).Value
        rcdNewRecSet.Fields(4).Value = rcdOldRecSet.Fields(4).Value
        rcdNewRecSet.Fields(5).Value = Primer
        rcdNewRecSet.Fields(6).Value = rcdOldRecSet.Fields(i).Name
        rcdNewRecSet.Fields(7).Value = rcdOldRecSet(i).Value
        rcdNewRecSet.Update
        End If
    Next i
    rcdOldRecSet.MoveNext
    j = j + 1
    varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
Wend
varReturnVal = SysCmd(acSysCmdClearStatus)
' turn off hourglass
DoCmd.Hourglass False

' reset status bar
varReturnVal = SysCmd(SYSCMD_CLEARSTATUS)


End Function

Function Create_bins(In_table As String, out_table As String)

' Declare variables used in this function
Dim tblOut_Table As TableDef
Dim rcdNewRecSet As Recordset, rcdOldRecSet As Recordset, intMsgRtn As Integer
Dim fldMyField As Field, varReturnVal As Variant

Dim oldtable As TableDef
Dim i As Integer, j As Integer
Dim NewF1 As String, NewF2 As String, NewF3 As String
Dim mmin, mmax As Integer
Dim NewT1 As Integer, NewT2 As Integer, NewT3 As Integer

Set cdb = CurrentDb()
' Initialize the cdb database variable to the current
' database
'Set cdb = DBEngine.Workspaces(0).Databases(0)

'If Exists(In_Table) = False Then Exit Function  'terminate
```

```
' Declaring new-field variables
NewF1 = cdb.TableDefs(In_table).Fields(0).Name
NewF2 = "Allele"
NewF3 = cdb.TableDefs(In_table).Fields(1).Name
' Declaring new-type variables
NewT1 = cdb.TableDefs(In_table).Fields(0).Type
NewT2 = DB_INTEGER
NewT3 = cdb.TableDefs(In_table).Fields(1).Type


'close all tables
On Error GoTo 0
For i = 0 To cdb.TableDefs.count - 1
   DoCmd.Close A_TABLE, cdb(i).Name
Next i

'update any changes
cdb.TableDefs.Refresh

'remove existing summary tables
If Delete_Tables(out_table) = False Then Exit Function  'terminate

Set tblOut_Table = cdb.CreateTableDef(out_table)     ' Creates new table
Set fldMyField = tblOut_Table.CreateField(NewF1, NewT1)  ' Creates a field in
Out_Table
tblOut_Table.Fields.Append fldMyField    ' Appends the NewF1 field to the fields
                          ' collection in the new table definition
Set fldMyField = tblOut_Table.CreateField(NewF2, NewT2)
tblOut_Table.Fields.Append fldMyField

Set fldMyField = tblOut_Table.CreateField(NewF3, NewT3)
tblOut_Table.Fields.Append fldMyField

' Append the new table to the TableDefs collection in the current database
cdb.TableDefs.Append tblOut_Table

Set rcdNewRecSet = cdb.OpenRecordset(out_table)

' Turn on hourglass
DoCmd.Hourglass True

Set rcdOldRecSet = cdb.OpenRecordset(In_table)
Set oldtable = cdb.TableDefs(In_table)


On Error GoTo 0
```

```
' initializes status bar counter
j = 0

' initializes the status bar
varReturnVal = SysCmd(SYSCMD_INITMETER, "Updating '" & In_table & " Table'",
20000)

' contents of the allele range table adding values between min max to Out_Table
While (Not rcdOldRecSet.EOF)
      varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
      mmin = rcdOldRecSet.Fields(2).Value
      mmax = rcdOldRecSet.Fields(3).Value
      For i = mmin To mmax
         rcdNewRecSet.AddNew
         rcdNewRecSet.Fields(NewF1) = rcdOldRecSet.Fields(NewF1).Value
         rcdNewRecSet.Fields(NewF3) = rcdOldRecSet.Fields(NewF3).Value
         rcdNewRecSet.Fields(NewF2) = i
         rcdNewRecSet.Update
         Next
         rcdOldRecSet.MoveNext
         j = j + 13
   varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
Wend

' turn off hourglass
DoCmd.Hourglass False

' reset status bar
varReturnVal = SysCmd(SYSCMD_CLEARSTATUS)


End Function

Private Function Delete_Tables(out_table)

'-This removes the table Out_Table if it exists.  The function prompts the user before
deleting the table
'.  It removes the relation object relating the Table_Name field of the
'table.  The subroutine terminates the program if either table is open or if
'an unrecognized error occurs (this will happen if a relation other than
'"Table_Name_Relation" exists which includes one of the tables).

'-The function returns true if it succeeded and false if it didn't.
```

```
    Dim i As Integer, msg As String
    On Error Resume Next

    'Test if tables already exist, prompt user if they do.
    For i = 0 To cdb.TableDefs.count - 1
       If (cdb(i).Name = out_table) Then
         msg = "OK to replace table """ & cdb(i).Name & """?"
         DoCmd.Hourglass False
         If MsgBox(msg, 4, Title) = 7 Then
           Delete_Tables = False
           Exit Function
         End If
         DoCmd.Hourglass True
       End If
    Next i

    'Ensure tables are closed.
    DoCmd.Close A_TABLE, out_table

    'Remove existing summary tables.  Err = 3265 occurs if tables don't exist.
    cdb.TableDefs.Delete out_table
    If Not (Err = 3265 Or Err = 0) Then GoTo MESSAGE:

    On Error GoTo 0
    Delete_Tables = True
    Exit Function   'normal subroutine exit


MESSAGE:    'show error message then quit
    MsgBox "An error has occurred." & Chr(13) & Chr(10) & "Access: " & Error, , Title
    Delete_Tables = False
    Exit Function

End Function

Private Function Exists(out_table)

'Checks to see if Out_Table Exists
'if an unrecognized error occurs (this will happen if a relation other than
""Table_Name_Relation" exists which includes one of the tables).

' The function returns true if it succeeded and false if it didn't.


    Dim i As Integer, msg As String
    On Error Resume Next
```

```
    'Test if tables already exist, prompt user if they do.
    For i = 0 To cdb.TableDefs.count - 1
       If (cdb(i).Name = out_table) Then
          Exists = True
          Exit Function   'normal subroutine exit
        End If
    Next i
    Exists = False
    Exit Function

MESSAGE:    'show error message then quit
    MsgBox "An error has occurred." & Chr(13) & Chr(10) & "Access: " & Error, , Title
    Exists = False
    Exit Function


End Function

Function CreateBaseTable(In_table As String, out_table As String, Primer As
String)
' will take baseline input and append together
' run from BaselineAppendMacro
' If reset = 1 delete table if reset = 0 Append to table
' Declare variables used in this function
Dim tblOut_Table As TableDef
Dim rcdNewRecSet As Recordset, rcdOldRecSet As Recordset, intMsgRtn As Integer
Dim fldMyField As Field, varReturnVal As Variant
Dim oldtable As TableDef, numfields As Integer
Dim i As Integer, j As Integer
Dim NewF1 As String, NewF2 As String, NewF3 As String, NewF4 As String
Dim NewF5 As String, NewF6 As String
Dim NewT1 As Integer, NewT2 As Integer, NewT3 As Integer, NewT4 As Integer
Dim NewT5 As Integer, NewT6 As Integer

' Initialize the cdb database variable to the current
' database
Set cdb = DBEngine.Workspaces(0).Databases(0)


' Declaring new-field and new-type variables
NewF1 = cdb.TableDefs(In_table).Fields(1).Name
NewF2 = cdb.TableDefs(In_table).Fields(2).Name
NewF3 = cdb.TableDefs(In_table).Fields(3).Name
NewF4 = "Locus"
NewF5 = "Allele"
```

```
NewF6 = "Value"

NewT1 = cdb.TableDefs(In_table).Fields(1).Type
NewT2 = cdb.TableDefs(In_table).Fields(2).Type
NewT3 = cdb.TableDefs(In_table).Fields(3).Type
NewT4 = DB_TEXT
NewT5 = DB_INTEGER
NewT6 = cdb.TableDefs(In_table).Fields(4).Type

'close all tables
On Error GoTo 0
For i = 0 To cdb.TableDefs.count - 1
   DoCmd.Close A_TABLE, cdb(i).Name
Next i

'update any changes
cdb.TableDefs.Refresh

'remove existing summary tables
If Delete_Tables(out_table) = False Then Exit Function  'terminate

Set tblOut_Table = cdb.CreateTableDef(out_table)     ' Creates new table
Set fldMyField = tblOut_Table.CreateField(NewF1, NewT1)  ' Creates a field in
Out_Table
tblOut_Table.Fields.Append fldMyField    ' Appends the NewF1 field to the fields
                         ' collection in the new table definition
Set fldMyField = tblOut_Table.CreateField(NewF2, NewT2)
tblOut_Table.Fields.Append fldMyField

Set fldMyField = tblOut_Table.CreateField(NewF3, NewT3)
tblOut_Table.Fields.Append fldMyField

Set fldMyField = tblOut_Table.CreateField(NewF4, NewT4)
tblOut_Table.Fields.Append fldMyField

Set fldMyField = tblOut_Table.CreateField(NewF5, NewT5)
tblOut_Table.Fields.Append fldMyField

Set fldMyField = tblOut_Table.CreateField(NewF6, NewT6)
tblOut_Table.Fields.Append fldMyField

' Append the new table to the TableDefs collection in the current database
cdb.TableDefs.Append tblOut_Table

Set rcdNewRecSet = cdb.OpenRecordset(out_table)
```

```
' Turn on hourglass
DoCmd.Hourglass True

Set rcdOldRecSet = cdb.OpenRecordset(In_table)
Set oldtable = cdb.TableDefs(In_table)
numfields = oldtable.Fields.count   ' number of columns in the CrossTabbed table

On Error GoTo 0

' initializes status bar counter
j = 0

' initializes the status bar
varReturnVal = SysCmd(SYSCMD_INITMETER, "Adding '" & In_table & " to
combined table'", 20000)

' copies the contents of the CrossTabbed table into Out_Table
While (Not rcdOldRecSet.EOF)
   For i = 4 To (numfields - 1)
      varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
      rcdNewRecSet.AddNew
      rcdNewRecSet.Fields(NewF1) = rcdOldRecSet.Fields(NewF1).Value
      rcdNewRecSet.Fields(NewF2) = rcdOldRecSet.Fields(NewF2).Value
      rcdNewRecSet.Fields(NewF3) = rcdOldRecSet.Fields(NewF3).Value
      rcdNewRecSet.Fields(NewF4) = Primer
      rcdNewRecSet.Fields(NewF5) = rcdOldRecSet.Fields(i).Name
      rcdNewRecSet.Fields(NewF6) = rcdOldRecSet(i).Value
      rcdNewRecSet.Update
   Next i
   rcdOldRecSet.MoveNext
   j = j + 2
   varReturnVal = SysCmd(SYSCMD_UPDATEMETER, j)
Wend

' turn off hourglass
DoCmd.Hourglass False

' reset status bar
varReturnVal = SysCmd(SYSCMD_CLEARSTATUS)


End Function

  Public Function bayes_mix()

     'This function creates a fixed-width text file using the spam mixture table
```

```
Dim cdb As Database
Dim rstGeno, rstMax, rstExp As Recordset
Dim ValStr As String, fishstr As String, jday As Integer
Dim i, j As Integer, formi, formj, Path, msg As String
Dim cnt As Integer, max_allele As String, varReturnVal As Variant
Dim calc_geno As String, gen_fish As Integer, gen_locus As String
Dim intfile
'Set some parameters
    Set cdb = CurrentDb()
    Set rstGeno = cdb.OpenRecordset("10_bayesmix(5)")
    Set rstMax = cdb.OpenRecordset("lociprams")
    Set rstExp = cdb.OpenRecordset("ExportSpecs")
    intfile = FreeFile      ' Get the # of next free file
Do While Not rstExp.EOF
  If rstExp![Application] = "bayes" Then
    Path = rstExp![output_path] & "bayes.mix"
  End If
  rstExp.MoveNext
Loop
  Open Path For Output As intfile


gen_fish = rstGeno!Fish ' initalize gen_fish with first value
gen_locus = rstGeno!locus
jday = rstGeno!jday

Do While Not rstGeno.EOF
    If gen_fish = rstGeno!Fish And jday = rstGeno!jday Then
      If rstGeno!locus = rstMax!locus Then
        If rstMax!HW_corr = "no" Then 'yes locus, no hw
          For i = 1 To rstMax!max_allele
          For j = i To rstMax!max_allele
            formi = Format(i, "00")
            formj = Format(j, "00")
            calc_geno = formi & formj
            If Not rstGeno.EOF Then
              If calc_geno = rstGeno!gen Then
                ValStr = ValStr + "1"
              Else
                ValStr = ValStr + "0"
              End If
            End If
          Next j
          Next i
            If Not rstGeno.EOF Then
              rstGeno.MoveNext
```

```
            End If
        Else
        i = 1
          For i = 1 To rstMax!allele 'yes locus, yes hw
            If Not rstGeno.EOF Then
              If i = rstGeno!allele And rstGeno!locus = rstMax!locus Then
                ValStr = ValStr & rstGeno!Value
                rstGeno.MoveNext
              Else
                ValStr = ValStr + "0"
              End If
            End If
          Next i
        End If
      Else
        If rstMax!HW_corr = "no" Then 'no locus, no hw
          For i = 1 To rstMax!max_allele
          For j = i To rstMax!max_allele
            ValStr = ValStr + "0"
          Next j
          Next i

        Else
          For i = 1 To rstMax!allele   'no locus, yes hw
            ValStr = ValStr + "0"
          Next i
        End If
      End If
        ValStr = ValStr + " "
        rstMax.MoveNext
Else
  If Not rstMax.EOF Then      'fill in when a fish has run out of loci
    If rstMax.HW_corr = "no" Then
      For i = 1 To rstMax!allele
      For j = i To rstMax!allele
        ValStr = ValStr + "0"
      Next j
      Next i
    Else
      For i = 1 To rstMax!allele
      ValStr = ValStr + "0"
      Next i
    End If
      ValStr = ValStr + " "
      rstMax.MoveNext
  End If
```

```
        End If

'criteria for printing out
        If rstMax.EOF Then
           Print #intfile, ValStr, gen_fish
           ValStr = ""
           rstMax.MoveFirst
           If Not rstGeno.EOF Then
              gen_fish = rstGeno!Fish
              jday = rstGeno!jday
           End If
        End If

Loop

 'close all tables
  Close #intfile
  varReturnVal = SysCmd(acSysCmdClearStatus)
  msg = Path & " " & " created"
  MsgBox msg, vbOKOnly
     End Function

    Public Function gda()
'outputs genpop table

Dim cdb As Database
Dim tbl As TableDef, varReturnVal As Variant
Dim rstGeno, rstMax, in2_sum, in3_sum, rstExp As Recordset
Dim fld As Field
Dim intfile
Dim g_len, f_len, y_len As Integer, alin As String
Dim cnt, ValStr, gen_fish, gen_locus, gen_year, gen_stock, Path, msg As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries

    Set cdb = CurrentDb()
    Set rstGeno = cdb.OpenRecordset("07_gda(3)")
    Set rstMax = cdb.OpenRecordset("lociprams")
    Set in2_sum = cdb.OpenRecordset("05_gda header(2)")
    Set in3_sum = cdb.OpenRecordset("05_gda header(3)")
    Set rstExp = cdb.OpenRecordset("ExportSpecs")
    intfile = FreeFile ' Get the # of next free file
 Do While Not rstExp.EOF
```

```
    If rstExp![Application] = "gda" Then
       Path = rstExp![output_path] & "gda.nex"
    End If
    rstExp.MoveNext
  Loop
    Open Path For Output As intfile

ValStr = "dimensions nloci=" & in3_sum!locus & " " & "npops=" & " " & in3_sum!stock
& ";"


  'Count the number of fish in each genotype and output to table by stock and locus

gen_locus = rstGeno!locus 'Assign first value for locus from max_alleles table
gen_fish = rstGeno!Fish
gen_stock = rstGeno![stock name]
  Print #intfile, "#nexus"
  Print #intfile, ""
  Print #intfile, "begin gdadata;"
  Print #intfile, ValStr
  Print #intfile, "     format missing=? separator=/;"
  Print #intfile, "     locusallelelabels"
  cnt = 1
  Do While Not in2_sum.EOF
    alin = Len(in2_sum!locus)
    Print #intfile, "        " & cnt & Space(10 - alin) & in2_sum!locus & ","
    in2_sum.MoveNext
    cnt = cnt + 1
  Loop
  Print #intfile, " ;"
  Print #intfile, "     matrix"

Do While Not StrComp(rstGeno![stock name], gen_stock)
    g_len = Len(rstGeno![stock name])
    Print #intfile, gen_stock & Space(g_len) & ":"
    Do While Not StrComp(gen_fish, rstGeno!Fish)     ' loop through each fish
    If Not rstGeno.EOF Then
       varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!Fish)
    End If

       f_len = Len(gen_fish)
       y_len = Len(rstGeno!Year)
       ValStr = rstGeno!Year & Space(4 - y_len) & "_" & gen_fish & Space(4 - f_len) & "
"
          Do While gen_fish = rstGeno!Fish And gen_year = rstGeno!Year
```

```
            If rstGeno!locus = rstMax!locus And Not rstMax.EOF Then
               cnt = rstGeno!Geno
               rstGeno.MoveNext
               If Not rstMax.EOF Then 'move max!locus down on record
                  rstMax.MoveNext
               End If
               ValStr = ValStr & " " & cnt ' concant. string of counts by stock/locus
            ElseIf rstGeno!locus > rstMax!locus Then   'move ahead one locus if no value
in data
                  rstMax.MoveNext
                  ValStr = ValStr & " " & " ?/? "
            ElseIf rstGeno!locus < rstMax!locus Then  ' fill in last last empty locus per fish
                  ValStr = ValStr & " " & " ?/? "
            End If
            If rstGeno.EOF Then
               GoTo dump
            End If
         Loop
dump:
         Do While Not rstMax.EOF      ' add ?/?
            ValStr = ValStr & " " & " ?/? "
            rstMax.MoveNext
         Loop
            Print #intfile, ValStr
            ValStr = ""
            If rstGeno.EOF Then
               GoTo Loopout
            End If

         If Not rstGeno.EOF Then
            gen_fish = rstGeno!Fish
            gen_year = rstGeno!Year
            rstMax.MoveFirst
         Else
            GoTo Loopout
         End If
         If gen_stock <> rstGeno![stock name] Then
         GoTo outside
         End If

   Loop
outside:
   gen_stock = rstGeno![stock name]
   Print #intfile, ","
Loop
```

```
Loopout:
Print #intfile, ";"
Print #intfile, "end;"
' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " " & "created"
     MsgBox msg
 'close all tables
  Close #intfile

     End Function

  Public Function gdamix()

'outputs genpop table

Dim cdb As Database
Dim tbl As TableDef, varReturnVal As Variant
Dim rstGeno, rstMax, in2_sum, in3_sum, rstExp As Recordset
Dim fld As Field
Dim intfile
Dim g_len, f_len, y_len As Integer, alin As String
Dim cnt, ValStr, gen_fish, gen_locus, gen_year, gen_stock, Path, msg As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries

   Set cdb = CurrentDb()
   Set rstGeno = cdb.OpenRecordset("07_gdamix(3)")
   Set rstMax = cdb.OpenRecordset("lociprams")
   Set in2_sum = cdb.OpenRecordset("05_gdamix header(2)")
   Set in3_sum = cdb.OpenRecordset("05_gdamix header(3)")
   Set rstExp = cdb.OpenRecordset("ExportSpecs")
   intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "gda" Then
      Path = rstExp![output_path] & "gdamix.nex"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile
```

```
ValStr = "dimensions nloci=" & in3_sum!locus & " " & "npops=" & " " & in3_sum!stock
& ";"


 'Count the number of fish in each genotype and output to table by stock and locus

gen_locus = rstGeno!locus 'Assign first value for locus from max_alleles table
gen_fish = rstGeno!Fish
gen_stock = rstGeno![stock name]
 Print #intfile, "#nexus"
 Print #intfile, ""
 Print #intfile, "begin gdadata;"
 Print #intfile, ValStr
 Print #intfile, "     format missing=? separator=/;"
 Print #intfile, "     locusallelelabels"
 cnt = 1
 Do While Not in2_sum.EOF
  alin = Len(in2_sum!locus)
  Print #intfile, "        " & cnt & Space(10 - alin) & in2_sum!locus & ","
  in2_sum.MoveNext
  cnt = cnt + 1
 Loop
 Print #intfile, " ;"
 Print #intfile, "     matrix"

Do While Not StrComp(rstGeno![stock name], gen_stock)
   g_len = Len(rstGeno![stock name])
   Print #intfile, gen_stock & Space(g_len) & ":"
   Do While Not StrComp(gen_fish, rstGeno!Fish)     ' loop through each fish
   If Not rstGeno.EOF Then
      varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!Fish)
   End If

      f_len = Len(gen_fish)
      y_len = Len(rstGeno!Year)
      ValStr = rstGeno!Year & Space(4 - y_len) & "_" & gen_fish & Space(4 - f_len) & "
"
         Do While gen_fish = rstGeno!Fish And gen_year = rstGeno!Year
           If rstGeno!locus = rstMax!locus And Not rstMax.EOF Then
              cnt = rstGeno!Geno
              rstGeno.MoveNext
              If Not rstMax.EOF Then 'move max!locus down on record
                 rstMax.MoveNext
              End If
              ValStr = ValStr & " " & cnt ' concant. string of counts by stock/locus
```

```
            ElseIf rstGeno!locus > rstMax!locus Then   'move ahead one locus if no value
in data
                rstMax.MoveNext
                ValStr = ValStr & " " & " ?/? "
            ElseIf rstGeno!locus < rstMax!locus Then  ' fill in last last empty locus per fish
                ValStr = ValStr & " " & " ?/? "
            End If
            If rstGeno.EOF Then
              GoTo dump
            End If
        Loop
dump:
        Do While Not rstMax.EOF      ' add ?/?
          ValStr = ValStr & " " & " ?/? "
          rstMax.MoveNext
        Loop
          Print #intfile, ValStr
          ValStr = ""
          If rstGeno.EOF Then
            GoTo Loopout
          End If

        If Not rstGeno.EOF Then
          gen_fish = rstGeno!Fish
          gen_year = rstGeno!Year
          rstMax.MoveFirst
        Else
          GoTo Loopout
        End If
        If gen_stock <> rstGeno![stock name  Then
        GoTo outside
        End If

   Loop
outside:
   gen_stock = rstGeno![stock name]
   Print #intfile, ","
Loop

Loopout:
Print #intfile, ";"
Print #intfile, "end;"
' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " " & "created"
```

```
    MsgBox msg
'close all tables
 Close #intfile

    End Function

Function Vax_base()
'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno As Recordset, rstMax As Recordset, rstHW As Recordset
Dim fld As Field
Dim intfile, numfields As Integer, numstr As String, numstr_all As String
Dim ValStr As String
Dim i, Fish As Integer, j As Integer, k As Integer, max_locus As String, gen_stock As
Integer
Dim formi As String, formj As String, cnt As Double, max_allele As String
Dim gen_locus As String, calc_geno As String, gen_geno As Integer, test As String
Dim gen1_locus As String, HW_correction As String
Dim freq, A(99) As Double
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
    Set cdb = CurrentDb()
    Set rstGeno = cdb.OpenRecordset("04_geno_sp(7)")
    Set rstMax = cdb.OpenRecordset("max_allele")
    intfile = FreeFile      ' Get the # of next free file

    Open "H:\beachamt\coho\access\accbaseboot.out" For Output As intfile



 'Count the number of genotypes for each locus and output at top of file

Do Until rstMax.EOF
  numfields = rstMax!max_allele
  For i = 1 To rstMax!max_allele
   numfields = numfields + rstMax!max_allele - i
  Next i
  numstr = numfields
  rstMax.MoveNext 'read max_allele
  numstr_all = numstr_all & " " & numstr
  numstr = 0
```

```
Loop
 Print #intfile, numstr_all


 'Output as either genotype counts or HW corrected frequencies

gen_locus = rstGeno!locus
gen1_locus = rstGeno!locus
gen_stock = rstGeno!stock
max_allele = rstGeno!maxallele
HW_correction = rstGeno!HW_correction

Do While Not rstGeno.EOF

   If HW_correction = "no" Then

   'OUTPUT OBSERVED VALUES
        For i = 1 To max_allele
        For j = i To max_allele
           formi = Format(i, "00")
           formj = Format(j, "00")
           calc_geno = formi & formj

           If Not rstGeno.EOF Then
              If calc_geno = rstGeno!Geno Then
                 cnt = rstGeno!cntgeno
                 rstGeno.MoveNext
              End If
           End If
           ValStr = ValStr & "  " & Format(cnt, "#00.00")
           cnt = 0
        Next j
     Next i
     Print #intfile, ValStr
     ValStr = ""


     Else
 'OUTPUT HW CORRECTED VALUES
    i = 1
     For j = 1 To 99 'initialize vector A to 0
        A(j) = 0
     Next j

     Do While StrComp(gen_locus, gen1_locus) = 0
        If i = rstGeno!allele Then
```

```
            A(i) = rstGeno!freq
            Fish = rstGeno!xfish
            rstGeno.MoveNext
          Else
            A(i) = 0
          End If
          i = i + 1
          If Not rstGeno.EOF Then
              gen1_locus = rstGeno!locus
            Else
             gen1_locus = "ZZZZ"
            End If
      Loop

      For i = 1 To max_allele
        For j = i To max_allele
          If i = j Then
            cnt = A(i) * A(j) * Fish
          Else
             cnt = 2 * A(i) * A(j) * Fish
          End If
          ValStr = ValStr & "  " & Format(cnt, "#00.00")  ' concant. string of counts by
stock/locus
          cnt = 0
        Next j
      Next i
      Print #intfile, ValStr
      ValStr = ""
    End If

      If Not rstGeno.EOF Then
        If gen_stock <> rstGeno!stock Then
          Print #intfile,
        End If
      End If

      If Not rstGeno.EOF Then
        gen_locus = rstGeno!locus
        gen_stock = rstGeno!stock
        max_allele = rstGeno!maxallele
        HW_correction = rstGeno!HW_correction
        gen1_locus = rstGeno!locus
      Else
        GoTo Loopout
      End If
Loop
```

```
Loopout:

' turn off hourglass
DoCmd.Hourglass False

  MsgBox "H:\beachamt\coho\access\baseboot.out created"
  Close #intfile
End Function

 Function Vax_mix()
'takes output query from 06_genotypemix(7) and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim rstGeno As Recordset, rstMax As Recordset
Dim ValStr As String, fishstr As String
Dim i As Integer, j As Integer, formi As String, formj As String
Dim cnt As Integer, max_allele As String
Dim calc_geno As String, gen_fish As Integer
Dim intfile
'Set some parameters
     Set cdb = CurrentDb()
     Set rstGeno = cdb.OpenRecordset("03_genotypemix(4)")
     Set rstMax = cdb.OpenRecordset("max_allele")
     intfile = FreeFile       ' Get the # of next free file

     Open "H:\beachamt\coho\access\acctest.out" For Output As intfile

gen_fish = rstGeno!Fish ' initalize gen_fish with first value

   Do While Not rstGeno.EOF   ' loop through each fish
      If rstGeno!locus = rstMax!locus And rstGeno!Fish = gen_fish Then
         For i = 1 To rstGeno!maxallele
            For j = i To rstGeno!maxallele
               formi = Format(i, "00")
               formj = Format(j, "00")
               calc_geno = formi & formj

            If Not rstGeno.EOF Then
             cnt = cnt + 1
               If calc_geno = rstGeno!Geno Then
                 rstGeno.MoveNext
                 ValStr = Format(cnt, "000")
                 GoTo NoMoreCounts
               End If
            End If
```

```
        Next j
      Next i
NoMoreCounts:
      Else
        ValStr = "  0"
      End If

      fishstr = fishstr & "  " & ValStr
      cnt = 0
      ValStr = ""

      If Not rstMax.EOF Then
       rstMax.MoveNext
      End If

      If Not rstGeno.EOF Then
        If gen_fish <> rstGeno!Fish And rstMax.EOF Then
          Print #intfile, fishstr
          fishstr = ""
          gen_fish = rstGeno!Fish
          rstMax.MoveFirst
        End If
      Else
      Print #intfile, fishstr
       GoTo Loopout
      End If
   Loop

Loopout:
 MsgBox "H:\beachamt\coho\access\test.out created"
 Close #intfile
End Function

 Function bayes_bse()
'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef, varReturnVal As Variant
Dim rstBay, rstLoci, rstExp As Recordset
Dim fld As Field
Dim intfile, numfields As Integer, numstr, numstr_all As String
Dim ValStr As String, l_len, n_len, s_len, cnt_len, Allele_Tot As Integer
Dim i, j, k, max_locus, stock_code, locus_code, gen_stock, num_tot As Integer
Dim formi, formj, cnt, max_allele, RetVal, msg, Path As String
Dim gen_locus, calc_geno As String, gen_geno As Integer, test As String
```

```
Dim gen1_locus, HW_correction, gen_name As String, alle_cnt As Integer
Dim freq, A(99) As Double, Fish As Integer, fishno As String, nam_str As String
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
   Set cdb = CurrentDb()
   Set rstBay = cdb.OpenRecordset("10_bayesbase(3)")
   Set rstLoci = cdb.OpenRecordset("lociprams")
   Set rstExp = cdb.OpenRecordset("ExportSpecs")
   intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "bayes" Then
      Path = rstExp![output_path] & "bayes.bse"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile


' Output as  HW corrected frequencies

max_allele = rstBay!max_allele
Allele_Tot = rstBay!alleTot
locus_code = rstBay!locus_code
stock_code = rstBay!stock


Do While Not rstBay.EOF

' NO OUTPUT FOR PHENOTYPES

  'OUTPUT HW CORRECTED VALUES
      s_len = Len(stock_code)
      l_len = Len(locus_code)
      n_len = Len(rstBay!alleTot)
      ValStr = Space(6 - s_len) & stock_code & Space(6 - l_len) & locus_code & Space(6
- n_len) & Allele_Tot & " "
      i = 1
      Do While StrComp(locus_code, rstBay!locus_code) = 0
      If Not rstBay.EOF Then
         varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock code=" &
rstBay![stock] & " Locus code=" & rstBay!locus_code)
      End If
         If Not rstBay.EOF Then
            For i = 1 To max_allele
```

```
            If Not rstBay.EOF Then
              If i = rstBay!allele And rstBay!locus_code = locus_code Then
                ValStr = ValStr & Format(rstBay!freq, "0.0000")
                rstBay.MoveNext
              Else
                ValStr = ValStr & "0.0000"
              End If
            End If
          Next i
        End If
        'If rstloci!active_loci = "yes" Then 'Print out only requested alleles
          Print #intfile, ValStr
          ValStr = ""
        'End If
        rstLoci.MoveNext
      If rstBay.EOF Then GoTo outloop

    Loop
outloop:


    If Not rstBay.EOF Then
      If stock_code <> rstBay!stock Then
        stock_code = rstBay![stock]
        rstLoci.MoveFirst
      End If
    End If

    If Not rstBay.EOF Then
      locus_code = rstBay!locus_code
      stock_code = rstBay!stock
      max_allele = rstBay!max_allele
      Allele_Tot = rstBay!alleTot
    Else
      GoTo Loopout
    End If
Loop
Loopout:

' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)

msg = Path & " " & "created"
MsgBox Path
 Close #intfile
```

```
End Function

Function lociprams()

Dim cdb As Database
Dim tbl As TableDef, varReturnVal As Variant
Dim rstbase, allele_cnt, RecSet As Recordset
Dim locus, old_locus, HW_corr, active_loci As String
Dim locus_code, i As Integer

' Turn on hourglass
DoCmd.Hourglass True
varReturnVal = SysCmd(acSysCmdSetStatus, "Building loci table....")

    Set cdb = CurrentDb()
    Set rstbase = cdb.OpenRecordset("11_allbase_sort")
    Set allele_cnt = cdb.OpenRecordset("11_manautobin_allele_cnts")

  On Error Resume Next
   DoCmd.DeleteObject acTable, "lociprams"

' CREATE LOCIPRAM TABLE
   Dim fld1, fld2, fld3, fld4, fld5, fld6 As Field
   Set cdb = CurrentDb
   Set tbl = cdb.CreateTableDef("lociprams")
   Set fld1 = tbl.CreateField("locus", dbText)
   Set fld2 = tbl.CreateField("locus_code", dbInteger)
   Set fld3 = tbl.CreateField("HW_corr", dbText)
   Set fld4 = tbl.CreateField("allele", dbInteger)
   Set fld5 = tbl.CreateField("bin_allele", dbInteger)

   tbl.Fields.Append fld1
   tbl.Fields.Append fld2
   tbl.Fields.Append fld3
   tbl.Fields.Append fld4
   tbl.Fields.Append fld5
   cdb.TableDefs.Append tbl
   cdb.TableDefs.Refresh

'FILL LOCIPRAMS TABLE
   Set RecSet = cdb.OpenRecordset("lociprams")
locus_code = 1
   Do While Not rstbase.EOF
      If StrComp(old_locus, rstbase!locus) = -1 Then
         RecSet.AddNew
         RecSet.Fields(0) = rstbase!locus
```

```
        RecSet.Fields(1) = locus_code
        RecSet.Fields(2) = "yes"
        If allele_cnt!locus = rstbase!locus Then
           RecSet.Fields(3) = allele_cnt!allele
           RecSet.Fields(4) = allele_cnt!bin_allele
           allele_cnt.MoveNext
        End If
        RecSet.Update
        locus_code = locus_code + 1
      End If
      old_locus = rstbase!locus
      rstbase.MoveNext
   Loop
' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
MsgBox "Loci Prameter table complete"
Call CheckAlleleCnts

'close all tables
On Error GoTo 0
For i = 0 To cdb.TableDefs.count - 1
   DoCmd.Close A_TABLE, cdb(i).Name
Next i


End Function

Function CheckAlleleCnts()

Dim cdb As Database
Dim AlleleCntExp, AlleleCnt As Recordset
'compare allele counts with
'what we expect from import table with what we have in manual bin table

Set cdb = CurrentDb()
Set AlleleCntExp = cdb.OpenRecordset("11_ImportSpecTbl_sort")
Set AlleleCnt = cdb.OpenRecordset("11_manautobin_allele_cnts")
Do While Not AlleleCntExp.EOF
   If AlleleCntExp!alleles <> AlleleCnt!allele Then
      MsgBox "Loci  " & AlleleCnt!locus & "  total number of alleles expected for
binning do not match"
   End If
   AlleleCnt.MoveNext
   AlleleCntExp.MoveNext
Loop
```

```
End Function

Function Bayes_ctl()

'This file outputs a Bayes_control file.

Dim cdb As Database
Dim tbl As TableDef
Dim rststock, rstskcode, rstLoci, rstExp As Recordset
Dim fld As Field
Dim intfile, cnt, slen, s_len, ch_len, r_len, gen_geno As Integer
Dim varstr, formi, locstr, hw_type, typlen, loclen, ilen, alelen, formj, Path, msg As String
Dim i, j, gen_locus, calc_geno, max_alle, region As Integer
Dim newcode, oldcode, old_allele As Integer
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
Set cdb = CurrentDb()
Set rstskcode = cdb.OpenRecordset("06_check regions(2)")
Set rststock = cdb.OpenRecordset("06_check regions(3)")
Set rstLoci = cdb.OpenRecordset("lociprams")
Set rstExp = cdb.OpenRecordset("ExportSpecs")
intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "bayes" Then
      Path = rstExp![output_path] & "bayes.ctl"
   End If
   rstExp.MoveNext
Loop
Open Path For Output As intfile

Print #intfile, "mm2.4"
Print #intfile, "bayes.bse"
Print #intfile, "bayes.mix"
Print #intfile, "bayes1.Sum"
Print #intfile, "bayes1.bot"
Print #intfile, "bayes1.frq"
Print #intfile, "bayes1.B01"
Print #intfile, 235
Print #intfile, rststock.stockcnt 'number of stocks

i = 0
Do While Not rstLoci.EOF        'number of loci
   i = i + 1
   rstLoci.MoveNext
```

```
Loop

j = 1                          ' number of regions
region = rstskcode!regcode
Do While Not rstskcode.EOF
   If region <> rstskcode!regcode Then
      j = j + 1
      region = rstskcode!regcode
   End If
   rstskcode.MoveNext
   If Not rstskcode.EOF Then

   End If
Loop

Print #intfile, j
Print #intfile, i
Print #intfile, 899271
Print #intfile, 4480026
Print #intfile, 90092812
Print #intfile, 1
Print #intfile, 1

rstLoci.MoveFirst    'format for mix fortran input
   Do While Not rstLoci.EOF
      varstr = varstr & "1x," & rstLoci!allele & "I1,"
      rstLoci.MoveNext
   Loop

Print #intfile, "(" & varstr & ")"

rstLoci.MoveFirst    'format for base fortran input
Do While Not rstLoci.EOF
   If old_allele < rstLoci!allele Then
      max_alle = rstLoci!allele
   End If
     old_allele = max_alle
     rstLoci.MoveNext
Loop


Print #intfile, "(3I6, 1x," & max_alle & "f6.4)"
Print #intfile, "F F F F"

rstLoci.MoveFirst
Do While Not rstLoci.EOF    'character definition block
```

```
   ch_len = Len(rstLoci!locus)
   Print #intfile, Format(rstLoci!locus_code, "000") & " " & Format(rstLoci!allele, "00")
& "  T  " & rstLoci!locus & Space(18 - ch_len)
   rstLoci.MoveNext
Loop

rstskcode.MoveFirst
j = 1
region = rstskcode!regcode
Do While Not rstskcode.EOF    'stock definition block
   s_len = Len(rstskcode!stock)
   r_len = Len(j)
   Print #intfile, Format(rstskcode!stkcode, "000") & "  " & Format(j, "###") & " " &
Format(1 / rststock!stockcnt, ".000000") & "  " & rstskcode!stock & Space(18 - s_len) &
" .010000"
   If Not rstskcode.EOF And region <> rstskcode!regcode Then
      j = 1 + j
      region = rstskcode!regcode
   End If
   rstskcode.MoveNext
Loop

 ' turn off hourglass
DoCmd.Hourglass False
msg = Path & " created"
 MsgBox msg
 'close all tables
 Close #intfile
End Function

 Function neural_bse()

'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno, rstPram As Recordset
Dim fld As Field
Dim intfile, numfields As Integer
Dim i, j, k, k_len, s_len, max_allele, old_stock, old_year, old_fish As Integer
Dim ValStr, formi, formj, calc_geno As String

' Turn on hourglass
DoCmd.Hourglass True
```

```
'Set some parameters
   Set cdb = CurrentDb()
   Set rstGeno = cdb.OpenRecordset("04_geno_nn(6)")
   Set rstPram = cdb.OpenRecordset("lociprams")
   intfile = FreeFile      ' Get the # of next free file

   Open "c:\02_mixtures\NNcoho.bse" For Output As intfile

'initalize values
   old_stock = rstGeno!stock
   old_year = rstGeno!Year
   old_fish = rstGeno!Fish

Do While Not rstGeno.EOF
   s_len = Len(rstGeno!stock)
   ValStr = Space(5 - s_len) & rstGeno!stock
   If rstGeno!stock = old_stock And rstGeno!Year = old_year And rstGeno!Fish =
old_fish Then
      Do While Not rstPram.EOF
         k = 0
         If Not rstGeno.EOF And rstGeno!locus = rstPram!locus And rstPram!active_loci
= "yes" Then
            For i = 1 To rstGeno!max_allele
               For j = i To rstGeno!max_allele
                  formi = Format(i, "00")
                  formj = Format(j, "00")
                  calc_geno = formi & formj
                     'If Not rstGeno.EOF Then
                        If calc_geno <= rstGeno!gen Then
                           k = k + 1
                           If calc_geno = rstGeno!gen Then
                              rstGeno.MoveNext
                              GoTo found
                           End If
                        End If
                     'End If
               Next j
            Next i
         Else
            k = 0 'no geno for this marker
         End If   'check with a loci
 found:
         k_len = Len(k)
         ValStr = ValStr & Space(4 - k_len) & k
         rstPram.MoveNext
      Loop
```

```
      Print #intfile, ValStr
      ValStr = ""
    End If
    If Not rstGeno.EOF Then
      old_stock = rstGeno!stock
      old_year = rstGeno!Year
      old_fish = rstGeno!Fish
    End If
    rstPram.MoveFirst
   Loop
' turn off hourglass
DoCmd.Hourglass False

  MsgBox "c:\mm\neural.bse created"
  Close #intfile
 End Function

Function selectbasestock()
Dim tbl As TableDef
Dim sel_tbl, agg_tbl As Recordset
Dim i As Integer, StrSQL As String

Set cdb = CurrentDb()
DoCmd.SetWarnings 0

    On Error Resume Next
    DoCmd.DeleteObject acTable, "all_base"

' CREATE AUTOBIN TABLE
    Dim fld1, fld2, fld3, fld4, fld5, fld6, fld7 As Field
    Set tbl = cdb.CreateTableDef("all_base")
    Set fld1 = tbl.CreateField("tray", dbInteger)
    Set fld2 = tbl.CreateField("stock", dbInteger)
    Set fld3 = tbl.CreateField("year", dbInteger)
    Set fld4 = tbl.CreateField("fish", dbInteger)
    Set fld5 = tbl.CreateField("Locus", dbText)
    Set fld6 = tbl.CreateField("Allele", dbInteger)
    Set fld7 = tbl.CreateField("Value", dbInteger)
    tbl.Fields.Append fld1
    tbl.Fields.Append fld2
    tbl.Fields.Append fld3
    tbl.Fields.Append fld4
    tbl.Fields.Append fld5
    tbl.Fields.Append fld6
    tbl.Fields.Append fld7
    cdb.TableDefs.Append tbl
```

```
  cdb.TableDefs.Refresh

For Each tbl In cdb.TableDefs
  If InStr(1, tbl.Name, "bse_", 1) = 1 Then

     StrSQL = "SELECT DISTINCTROW " & tbl.Name & ".* INTO select_tbl" & _
" FROM " & tbl.Name & " INNER JOIN (Names LEFT JOIN main_aggregation ON
Names.stock " & _
" = main_aggregation.stock) ON " & tbl.Name & ".stock = Names.stock" & _
" WHERE (((main_aggregation.[Special Name])=[Forms]![MainMenu]![special]));"

DoCmd.RunSQL StrSQL
       Call Append_Tables("select_tbl", "all_base", Mid(CStr(tbl.Name), 5, 6))
  End If
 Next tbl
DoCmd.SetWarnings -1

'build new locipram tbl
Call lociprams
DoCmd.Close acTable, "temp"
End Function
Function selectmixstock()
Dim tbl As TableDef
Dim sel_tbl, agg_tbl, rsttemp As Recordset
Dim i As Integer, StrSQL, alcostr, cogastr As String
Dim fld1, fld2, fld3, fld4, fld5, fld6, fld7, fld8 As Field
Set cdb = CurrentDb()
Set rsttemp = cdb.OpenRecordset("temp")


DoCmd.SetWarnings 0

   On Error Resume Next
   DoCmd.DeleteObject acTable, "all_mix"

   'add julian date to mhc markers

 alcostr = "UPDATE mix_alco INNER JOIN mix_oki1 ON" & _
   "(mix_alco.stock = mix_oki1.stock) AND (mix_alco.fish = mix_oki1.fish)" & _
   "SET mix_alco.year = [mix_oki1].[year];"
 DoCmd.RunSQL alcostr

 cogastr = "UPDATE mix_coga INNER JOIN mix_oki1 ON" & _
   "(mix_coga.stock = mix_oki1.stock) AND (mix_coga.fish = mix_oki1.fish)" & _
   "SET mix_coga.year = [mix_oki1].[year];"
 DoCmd.RunSQL cogastr
```

```
If rsttemp!species = "chinook" Then
 'Build mixed table for chinook

   Set tbl = cdb.CreateTableDef("all_mix")
   Set fld1 = tbl.CreateField("tray", dbInteger)
   Set fld2 = tbl.CreateField("stock", dbInteger)
   Set fld3 = tbl.CreateField("year", dbInteger)
   Set fld4 = tbl.CreateField("jday", dbInteger)
   Set fld5 = tbl.CreateField("fish", dbInteger)
   Set fld6 = tbl.CreateField("Locus", dbText)
   Set fld7 = tbl.CreateField("Allele", dbInteger)
   Set fld8 = tbl.CreateField("Value", dbInteger)
   tbl.Fields.Append fld1
   tbl.Fields.Append fld2
   tbl.Fields.Append fld3
   tbl.Fields.Append fld4
   tbl.Fields.Append fld5
   tbl.Fields.Append fld6
   tbl.Fields.Append fld7
   tbl.Fields.Append fld8
   cdb.TableDefs.Append tbl
   cdb.TableDefs.Refresh

   For Each tbl In cdb.TableDefs
      If InStr(1, tbl.Name, "mix_", 1) = 1 Then

      StrSQL = "SELECT DISTINCTROW " & tbl.Name & ".* INTO select_tbl" & _
" FROM " & tbl.Name & " INNER JOIN [main_aggregate mix] ON " & tbl.Name &
".stock " & _
" = [main_aggregate mix].code WHERE ((([main_aggregate mix].[Special Namemix])"
& _
" = [Forms]![MainMenu]![specialmix]));"

DoCmd.RunSQL StrSQL

         Call Append_Tablesyr("select_tbl", "all_mix", Mid(CStr(tbl.Name), 5, 6))
      End If
   Next tbl
Else
' Build mixed table for non chinook
   Set tbl = cdb.CreateTableDef("all_mix")
   Set fld1 = tbl.CreateField("tray", dbInteger)
   Set fld2 = tbl.CreateField("stock", dbInteger)
   Set fld3 = tbl.CreateField("year", dbInteger)
   Set fld4 = tbl.CreateField("fish", dbInteger)
```

```
  Set fld5 = tbl.CreateField("Locus", dbText)
  Set fld6 = tbl.CreateField("Allele", dbInteger)
  Set fld7 = tbl.CreateField("Value", dbInteger)
  tbl.Fields.Append fld1
  tbl.Fields.Append fld2
  tbl.Fields.Append fld3
  tbl.Fields.Append fld4
  tbl.Fields.Append fld5
  tbl.Fields.Append fld6
  tbl.Fields.Append fld7
  cdb.TableDefs.Append tbl
  cdb.TableDefs.Refresh

  For Each tbl In cdb.TableDefs
    If InStr(1, tbl.Name, "mix_", 1) = 1 Then

    StrSQL = "SELECT DISTINCTROW " & tbl.Name & ".* INTO select_tbl" & _
" FROM " & tbl.Name & " INNER JOIN [main_aggregate mix] ON " & tbl.Name &
".stock " & _
" = [main_aggregate mix].code WHERE ((([main_aggregate mix].[Special Namemix])"
& _
" = [Forms]![MainMenu]![specialmix]));"

DoCmd.RunSQL StrSQL

      Call Append_Tables("select_tbl", "all_mix", Mid(CStr(tbl.Name), 5, 6))
    End If
  Next tbl
End If
DoCmd.SetWarnings -1
End Function

Function gclass_bse()

'updated nov29/2000
Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno, rstExp, in2_sum As Recordset
Dim fld As Field, varReturnVal As Variant
Dim intfile
Dim g_len, f_len, y_len, MinCnt As Integer
Dim gen_stock, gen_fish, gen_year As Integer
Dim cnt, ValStr, gen_locus, msg, Path As String

' Turn on hourglass
DoCmd.Hourglass True
```

```
'Open queries

Set cdb = CurrentDb()
MinCnt = Forms![01_GeneClassScrn]![locinumber]
Set rstGeno = cdb.OpenRecordset("07_genepop(1)")
Set in2_sum = cdb.OpenRecordset("05_gda header(2)")
Set rstExp = cdb.OpenRecordset("ExportSpecs")
intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "geneclass" Then
      Path = rstExp![output_path] & "geneclass.bse"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile

'Header information
Print #intfile, "geneclass for baseline data from MM 2.4"
Do While Not in2_sum.EOF
   Print #intfile, in2_sum!locus
   in2_sum.MoveNext
   cnt = cnt + 1
Loop

'Count the number of fish in each genotype and output to table by stock and locus
in2_sum.MoveFirst
gen_locus = rstGeno!locus
gen_stock = rstGeno![stock name]
gen_fish = rstGeno!Fish
gen_year = rstGeno!Year

Print #intfile, "POP"
Do While Not StrComp(rstGeno![stock name], gen_stock)
   Do While Not StrComp(gen_fish, rstGeno!Fish)
      g_len = Len(rstGeno![stock name])
      f_len = Len(gen_fish)
      y_len = Len(rstGeno!Year)
      If Not rstGeno.EOF Then
         varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!Fish)
      End If
      ValStr = rstGeno![stock name] & Space(18 - g_len) & rstGeno!Year & Space(5 -
y_len) & gen_fish & Space(4 - f_len) & " , "
      If rstGeno!lociCnt < MinCnt Then
         rstGeno.MoveNext
```

```
        GoTo dmptwo
      End If
      Do While gen_fish = rstGeno!Fish And gen_year = rstGeno!Year
        If rstGeno!locus = in2_sum!locus And Not in2_sum.EOF Then
          cnt = rstGeno!Geno
          rstGeno.MoveNext
          If Not in2_sum.EOF Then
            in2_sum.MoveNext
          End If
            ValStr = ValStr & " " & cnt
          ElseIf rstGeno!locus > in2_sum!locus Then
            in2_sum.MoveNext
            ValStr = ValStr & " " & "0000"
          ElseIf rstGeno!locus < in2_sum!locus Then
            ValStr = ValStr & " " & "0000"
          End If
        If rstGeno.EOF Then
          GoTo dump
        End If
      Loop

dump:
      Do While Not in2_sum.EOF 'fill in zero for a fish
        ValStr = ValStr & " " & "0000"
        in2_sum.MoveNext
      Loop

      Print #intfile, ValStr
dmptwo:
      ValStr = ""
      If Not rstGeno.EOF Then
        gen_fish = rstGeno!Fish
        gen_year = rstGeno!Year
        in2_sum.MoveFirst
      Else
        GoTo outside
      End If
      If gen_stock <> rstGeno![stock name] Then
        GoTo jumpout
      End If
    Loop
jumpout:
    gen_stock = rstGeno![stock name]
    Print #intfile, "POP"
Loop
outside:
```

```
varReturnVal = SysCmd(acSysCmdClearStatus)
DoCmd.Hourglass False
msg = Path & " created"
MsgBox msg
Close #intfile
End Function

Function structure()
Dim rstExp, rstData, rstLoci As Recordset
Dim fld As Field, varReturnVal As Variant
Dim intfile
Dim s_len, f_len, y_len As Integer
Dim gen_stock, cnt_stock, gen_fish, gen_year, LociCount, i As Integer
Dim cnt, ValStrA, ValStrB, gen_locus, msg, pop, Path As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries
Set cdb = CurrentDb()
Set rstExp = cdb.OpenRecordset("ExportSpecs")
Set rstData = cdb.OpenRecordset("07_gda(2b)")
Set rstLoci = cdb.OpenRecordset("lociprams")
LociCount = Forms![01_StructureBaseScrn]![ComboStructureScrn]
intfile = FreeFile ' Get the # of next free file

Do While Not rstExp.EOF
   If rstExp![Application] = "structure" Then
      Path = rstExp![output_path] & "structure.txt"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile
i = 0
pop = 0
Do While Not rstData.EOF
   If rstData!lociCnt < Int(LociCount) Then GoTo StrucOut
      gen_stock = Format(rstData!stock, "000")
      gen_year = rstData!Year
      gen_fish = Format(rstData!Fish, "0000")
      s_len = Len(gen_stock)
      y_len = Len(gen_year)
      f_len = Len(gen_fish)
      If cnt_stock <> rstData!stock Then
         pop = pop + 1
      End If
```

```
    cnt_stock = rstData!stock
    ValStrA = gen_stock & gen_year & gen_fish & Space(14 - (s_len + y_len + f_len))
& " " & pop & " 1"
    ValStrB = gen_stock & gen_year & gen_fish & Space(14 - (s_len + y_len + f_len))
& " " & pop & " 1"
    i = i + 1
    Do While gen_stock = Format(rstData!stock, "000") And gen_year = rstData!Year
And gen_fish = Format(rstData!Fish, "0000")
      If rstData!locus = rstLoci!locus Then
        ValStrA = ValStrA & " " & rstData!A
        ValStrB = ValStrB & " " & rstData!B
        rstData.MoveNext
      Else
        ValStrA = ValStrA & " -9"
        ValStrB = ValStrB & " -9"
      End If
        rstLoci.MoveNext
      If rstData.EOF Then ' check for data file end
        GoTo outloop
      End If
    Loop
outloop:
    Do While Not rstLoci.EOF  ' fill in missing loci
      ValStrA = ValStrA & " -9"
      ValStrB = ValStrB & " -9"
      rstLoci.MoveNext
    Loop
    Print #intfile, ValStrA
    Print #intfile, ValStrB
    rstLoci.MoveFirst
StrucOut:
  If rstData.EOF Then GoTo outloop2
    If rstData!lociCnt < Int(LociCount) Then
      rstData.MoveNext
    End If
Loop
outloop2:
DoCmd.Hourglass False
Close #intfile
msg = Path & " created" & " and number of fish in sample =" & i
MsgBox msg

End Function

Function structuremix()
Dim rstExp, rstData, rstLoci As Recordset
```

```
Dim fld As Field, varReturnVal As Variant
Dim intfile
Dim s_len, f_len, y_len As Integer
Dim gen_stock, gen_fish, gen_year, i As Integer
Dim cnt, ValStrA, ValStrB, gen_locus, msg, Path As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries
Set cdb = CurrentDb()
Set rstExp = cdb.OpenRecordset("ExportSpecs")
Set rstData = cdb.OpenRecordset("07_geneclassmix(2)")
Set rstLoci = cdb.OpenRecordset("lociprams")

intfile = FreeFile ' Get the # of next free file

Do While Not rstExp.EOF
   If rstExp![Application] = "structure" Then
      Path = rstExp![output_path] & "structuremix.txt"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile
i = 0
Do While Not rstData.EOF
   gen_stock = Format(rstData!stock, "0000")
   gen_year = rstData!Year
   gen_fish = Format(rstData!Fish, "0000")
   s_len = Len(gen_stock)
   y_len = Len(gen_year)
   f_len = Len(gen_fish)
   ValStrA = gen_stock & "_" & gen_year & "_" & gen_fish & Space(14 - (s_len + y_len
+ f_len)) & gen_stock & Space(5 - s_len) '& "0"
   ValStrB = gen_stock & "_" & gen_year & "_" & gen_fish & Space(14 - (s_len + y_len
+ f_len)) & gen_stock & Space(5 - s_len) '& "0"
   i = i + 1
   Do While gen_stock = Format(rstData!stock, "0000") And gen_year = rstData!Year
And gen_fish = Format(rstData!Fish, "0000")
      If rstData!locus = rstLoci!locus Then
         ValStrA = ValStrA & " " & rstData!A
         ValStrB = ValStrB & " " & rstData!B
         rstData.MoveNext
      Else
         ValStrA = ValStrA & " -9"
         ValStrB = ValStrB & " -9"
```

```
      End If
      rstLoci.MoveNext
      If rstData.EOF Then ' check for data file end
        GoTo outloop
      End If
    Loop
outloop:
    Do While Not rstLoci.EOF  ' fill in missing loci
      ValStrA = ValStrA & " -9"
      ValStrB = ValStrB & " -9"
      rstLoci.MoveNext
    Loop

    Print #intfile, ValStrA
    Print #intfile, ValStrB
    rstLoci.MoveFirst
Loop

DoCmd.Hourglass False
Close #intfile
msg = Path & " created" & " and number of fish in sample =" & i
MsgBox msg

End Function

Function spam_bsefreqs()
'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef
Dim rstMax, rstExp, rstfreq As Recordset
Dim fld As Field, varReturnVal As Variant
Dim numstr As String, numstr_all As String
Dim ValStr, msg, max_locus, test As String
Dim i, j, k, gen_stock, g_len, f_len, cnt_len, intfile, Totfish, numfields, Fish As Integer
Dim formi, formj, cnt, max_allele, RetVal, Path As String
Dim gen_locus, calc_geno As String, gen_geno As Integer
Dim gen1_locus, gen_name As String, alle_cnt As Integer
Dim freq As Double, A(99) As Double, nam_str As String
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
    Set cdb = CurrentDb()
    Set rstfreq = cdb.OpenRecordset("04_geno_sp(7)")
```

```
    Set rstMax = cdb.OpenRecordset("lociprams")
    Set rstExp = cdb.OpenRecordset("ExportSpecs")
    intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
  If rstExp![Application] = "spam" Then
    Path = rstExp![output_path] & "spamfreq.bse"
  End If
  rstExp.MoveNext
Loop
  Open Path For Output As intfile

gen_locus = rstfreq!locus
gen_stock = rstfreq!stock
max_allele = rstfreq!maxallele
gen_name = rstfreq![stock name]
Totfish = rstfreq!xfish

nam_str = "#" & " " & Format(gen_stock, "000") & " " & Format(gen_name,
"00000000000")
Print #intfile, nam_str

Do While Not rstfreq.EOF
  If Not rstfreq.EOF Then
    varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstfreq![stock
name] & " Locus=" & rstfreq![locus])
  End If
  g_len = Len(gen_locus)
  f_len = Len(Totfish)
  ValStr = gen_locus & Space(8 - g_len) & Totfish & Space(5 - f_len)
  i = 1
  Do While StrComp(gen_locus, rstfreq!locus) = 0
    If Not rstfreq.EOF Then
      For i = 1 To max_allele
        If Not rstfreq.EOF Then
          If i = rstfreq!allele And rstfreq!locus = gen_locus Then
            ValStr = ValStr & Format(rstfreq!freq, "0.0000")
            rstfreq.MoveNext
          Else
            ValStr = ValStr & Format(Sqr((((1 / max_allele) + 1) * (1 / max_allele)) /
((2 * Totfish + 2) * (2 * Totfish + 1))), "0.0000") 'add sample size corrected value
          End If
        End If
      Next i
    End If
    Print #intfile, ValStr
    ValStr = ""
```

```
      If rstfreq.EOF Then GoTo outloop
    Loop
outloop:
    If Not rstfreq.EOF Then
      If gen_stock <> rstfreq!stock Then
        Print #intfile,
        gen_name = rstfreq![stock name]
        gen_stock = rstfreq!stock
        nam_str = "#" & " " & Format(gen_stock, "000") & " " & Format(gen_name,
"00000000000")
        Print #intfile, nam_str
      End If
    End If
    If Not rstfreq.EOF Then
      gen_locus = rstfreq!locus
      gen_stock = rstfreq!stock
      max_allele = rstfreq!maxallele
      gen_name = rstfreq![stock name]
      Totfish = rstfreq![xfish]
    Else
    GoTo Loopout
    End If
Loop
Loopout:

' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " " & " created"
MsgBox msg, vbOKOnly

Close #intfile
End Function

 Function PCA_bsefreqs()
'takes output tables from 06_genotype table and max alleles query and
'outputs a table with all the genotype filled in

Dim cdb As Database
Dim tbl As TableDef
Dim rstMax, rstExp, rstfreq As Recordset
Dim fld As Field, varReturnVal As Variant
Dim numstr As String, numstr_all As String
Dim ValStr, msg, max_locus, test As String
Dim i, j, k, gen_stock, g_len, f_len, cnt_len, intfile, Totfish, numfields, Fish As Integer
Dim formi, formj, cnt, max_allele, RetVal, Path As String
```

```
Dim gen_locus, calc_geno As String, gen_geno As Integer
Dim gen1_locus, gen_name As String, alle_cnt As Integer
Dim freq As Double, A(99) As Double, nam_str As String
' Turn on hourglass
DoCmd.Hourglass True

'Set some parameters
    Set cdb = CurrentDb()
    Set rstfreq = cdb.OpenRecordset("05_phylip(3)")
    Set rstMax = cdb.OpenRecordset("lociprams")
    Set rstExp = cdb.OpenRecordset("ExportSpecs")
    intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
  If rstExp![Application] = "phylip" Then
    Path = rstExp![output_path] & "PCAfreq.bse"
  End If
  rstExp.MoveNext
Loop
  Open Path For Output As intfile

gen_locus = rstfreq!locus
gen_stock = rstfreq!stock
max_allele = rstfreq!maxunbin
gen_name = rstfreq![stock name]
Totfish = rstfreq!xfish

nam_str = Format(gen_name, "00000000000")
ValStr = nam_str
Do While Not rstfreq.EOF
  If Not rstfreq.EOF Then
    varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstfreq![stock
name] & " Locus=" & rstfreq![locus])
  End If
  g_len = Len(gen_locus)
  f_len = Len(Totfish)
  i = 1
  Do While StrComp(gen_locus, rstfreq!locus) = 0
    If Not rstfreq.EOF Then
      For i = 1 To max_allele
        If Not rstfreq.EOF Then
          If i = rstfreq!allele And rstfreq!locus = gen_locus Then
            ValStr = ValStr & " " & Format(rstfreq!freq, "0.000")
            rstfreq.MoveNext
          Else
            ValStr = ValStr & " 0.000"
          End If
```

```
            End If
          Next i
        End If
        'Print #intfile, ValStr
        'ValStr = ""
        If rstfreq.EOF Then GoTo outloop
      Loop
outloop:
      If Not rstfreq.EOF Then
        If gen_stock <> rstfreq!stock Then
          Print #intfile, ValStr
          ValStr = ""
          gen_name = rstfreq![stock name]
          gen_stock = rstfreq!stock
          ValStr = Format(gen_name, "00000000000")
        End If
      End If
      If Not rstfreq.EOF Then
        gen_locus = rstfreq!locus
        gen_stock = rstfreq!stock
        max_allele = rstfreq!maxunbin
        gen_name = rstfreq![stock name]
        Totfish = rstfreq![xfish]
      Else
      GoTo Loopout
      End If
Loop
Loopout:

' turn off hourglass
DoCmd.Hourglass False
varReturnVal = SysCmd(acSysCmdClearStatus)
msg = Path & " " & " created"
MsgBox msg, vbOKOnly

Close #intfile
End Function

Function gclass_mix()

'updated nov29/2000
Dim cdb As Database
Dim tbl As TableDef
Dim rstGeno, in2_sum, rstExp As Recordset
Dim fld As Field, varReturnVal As Variant
Dim intfile
```

```
Dim g_len, f_len, y_len As Integer
Dim gen_stock, gen_fish, gen_year, MinCnt As Integer
Dim cnt, ValStr, gen_locus, msg, Path As String

' Turn on hourglass
DoCmd.Hourglass True

'Open queries
Set cdb = CurrentDb()
MinCnt = Forms![01_GeneClassScrnmix]![locinumber]
Set rstGeno = cdb.OpenRecordset("07_geneclassmix(3)")
Set in2_sum = cdb.OpenRecordset("05_gda header(2)")
   Set rstExp = cdb.OpenRecordset("ExportSpecs")
   intfile = FreeFile ' Get the # of next free file
Do While Not rstExp.EOF
   If rstExp![Application] = "geneclass" Then
      Path = rstExp![output_path] & "geneclass.mix"
   End If
   rstExp.MoveNext
Loop
   Open Path For Output As intfile
'Header information
Print #intfile, "geneclass file from MM 2.3"
Do While Not in2_sum.EOF
   Print #intfile, in2_sum!locus
   in2_sum.MoveNext
   cnt = cnt + 1
Loop

in2_sum.MoveFirst
gen_locus = rstGeno!locus
gen_stock = rstGeno![stock name]
gen_fish = rstGeno!Fish
gen_year = rstGeno!Year

Print #intfile, "POP"
Do While Not StrComp(rstGeno![stock name], gen_stock)
   Do While Not StrComp(gen_fish, rstGeno!Fish)
      g_len = Len(rstGeno![stock name])
      f_len = Len(gen_fish)
      y_len = Len(rstGeno!Year)
      If Not rstGeno.EOF Then
         varReturnVal = SysCmd(acSysCmdSetStatus, "Output Stock=" & rstGeno![stock
name] & " year=" & rstGeno!Year & " fish=" & rstGeno!Fish)
      End If
```

```
      ValStr = rstGeno![stock name] & Space(18 - g_len) & rstGeno!Year & Space(5 -
y_len) & gen_fish & Space(4 - f_len) & " , "
      If rstGeno!lociCnt < MinCnt Then
         rstGeno.MoveNext
         GoTo dmptwo
       End If
      Do While gen_fish = rstGeno!Fish And gen_year = rstGeno!Year
         If rstGeno!locus = in2_sum!locus And Not in2_sum.EOF Then
            cnt = rstGeno!Geno
            rstGeno.MoveNext
            If Not in2_sum.EOF Then
               in2_sum.MoveNext
            End If
               ValStr = ValStr & " " & cnt
            ElseIf rstGeno!locus > in2_sum!locus Then
               in2_sum.MoveNext
               ValStr = ValStr & " " & "0000"
            ElseIf rstGeno!locus < in2_sum!locus Then
               ValStr = ValStr & " " & "0000"
            End If
         If rstGeno.EOF Then
            GoTo dump
         End If
      Loop

dump:
      Do While Not in2_sum.EOF 'fill in zero for a fish
         ValStr = ValStr & " " & "0000"
         in2_sum.MoveNext
      Loop

      Print #intfile, ValStr
dmptwo:
      ValStr = ""
      If Not rstGeno.EOF Then
         gen_fish = rstGeno!Fish
         gen_year = rstGeno!Year
         in2_sum.MoveFirst
      Else
         GoTo outside
      End If
      If gen_stock <> rstGeno![stock name] Then
         GoTo jumpout
      End If
   Loop
jumpout:
```

```
    gen_stock = rstGeno![stock name]
    Print #intfile, "POP"
  Loop
  outside:
  varReturnVal = SysCmd(acSysCmdClearStatus)
  DoCmd.Hourglass False
  msg = Path & " created"
  MsgBox msg
  Close #intfile
  End Function

    Public Function spam_mix()

      'This function creates a fixed-width text file using the spam mixture table
   Dim cdb As Database
  Dim rstGeno, rstMax, rstExp As Recordset
  Dim ValStr As String, fishstr As String, jday As Integer
  Dim i, j As Integer, formi, formj, Path, msg As String
  Dim cnt As Integer, max_allele As String, varReturnVal As Variant
  Dim calc_geno As String, gen_fish As Integer, gen_locus As String
  Dim intfile
  'Set some parameters
       Set cdb = CurrentDb()
       Set rstGeno = cdb.OpenRecordset("04_genomix_sp(5)")
       Set rstMax = cdb.OpenRecordset("lociprams")
       Set rstExp = cdb.OpenRecordset("ExportSpecs")
       intfile = FreeFile      ' Get the # of next free file
   Do While Not rstExp.EOF
     If rstExp![Application] = "spam" Then
        Path = rstExp![output_path] & "spam.mix"
     End If
     rstExp.MoveNext
  Loop
     Open Path For Output As intfile



  Print #intfile, "\"


  gen_fish = rstGeno!Fish ' initalize gen_fish with first value
  gen_locus = rstGeno!locus
  jday = rstGeno!jday

  Do While Not rstGeno.EOF
       If gen_fish = rstGeno!Fish And jday = rstGeno!jday Then
```

```
If rstGeno!locus = rstMax!locus Then
  If rstMax!HW_corr = "no" Then 'yes locus, no hw
    For i = 1 To rstMax!max_allele
    For j = i To rstMax!max_allele
      formi = Format(i, "00")
      formj = Format(j, "00")
      calc_geno = formi & formj
      If Not rstGeno.EOF Then
        If calc_geno = rstGeno!gen Then
          ValStr = ValStr + "1"
        Else
          ValStr = ValStr + "0"
        End If
      End If
    Next j
    Next i
      If Not rstGeno.EOF Then
        rstGeno.MoveNext
      End If
  Else
  i = 1
    For i = 1 To rstMax!bin_allele 'yes locus, yes hw
      If Not rstGeno.EOF Then
        If i = rstGeno!allele And rstGeno!locus = rstMax!locus Then
          ValStr = ValStr & rstGeno!Value
          rstGeno.MoveNext
        Else
          ValStr = ValStr + "0"
        End If
      End If
    Next i
  End If
Else
  If rstMax!HW_corr = "no" Then 'no locus, no hw
    For i = 1 To rstMax!max_allele
    For j = i To rstMax!max_allele
      ValStr = ValStr + "0"
    Next j
    Next i

  Else
    For i = 1 To rstMax!bin_allele  'no locus, yes hw
      ValStr = ValStr + "0"
    Next i
  End If
End If
```

```
            ValStr = ValStr + " "
            rstMax.MoveNext
        Else
          If Not rstMax.EOF Then      'fill in when a fish has run out of loci
            If rstMax.HW_corr = "no" Then
              For i = 1 To rstMax!bin_allele
              For j = i To rstMax!bin_allele
                ValStr = ValStr + "0"
              Next j
              Next i
            Else
              For i = 1 To rstMax!bin_allele
              ValStr = ValStr + "0"
              Next i
            End If
              ValStr = ValStr + " "
              rstMax.MoveNext
          End If
        End If

'criteria for printing out
        If rstMax.EOF Then
          Print #intfile, ValStr, gen_fish
          ValStr = ""
          rstMax.MoveFirst
          If Not rstGeno.EOF Then
            gen_fish = rstGeno!Fish
            jday = rstGeno!jday
          End If
        End If

Loop



 'close all tables
  Close #intfile
  varReturnVal = SysCmd(acSysCmdClearStatus)
  msg = Path & " " & " created"
  MsgBox msg, vbOKOnly
     End Function
```

Appendix D: Wood Algorithm for binning alleles.

Variables:

F - user input minimum bin frequency
p - user selection of baseline populations
k - total number of alleles per locus
f_{ij} - frequency of allele i at locus j across all populations p where $i = 1, k$ and $j = 1, \infty$ and $p = 1, n$.

The following is a series of steps the algorithm uses to select allele frequencies below user input threshold and combine these low frequency alleles with adjacent alleles.
After pooling across all selected populations, for each locus do: